MAGISTERARBEIT

im Studiengang Internationales Informationsmanagement
an der Universität Hildesheim

Modularisierung des Retrievalprozesses zur funktionellen Integration heterogener IR-Komponenten

Jan Hendrik Scheufen

Hildesheim 2002

Vorwort

Die vorliegende Arbeit bildet den Abschluss meines Studiums an der Universität Hildesheim, wo ich seit 1996 im Magisterstudiengang 'Internationales Informationsmanagement' eingeschrieben bin.

Zu Beginn meiner Studienzeit waren Schlagwörter wie *World Wide Web* und *Information Highway* gerade in aller Munde und das WWW war im Begriff, den Einfluss der Informations- und Kommunikationstechnologie auf viele Bereiche des modernen Lebens auszudehnen. Seither sind die Annehmlichkeiten und auch die Missstände des 'Netzes der Netze' für viele von uns zu einer alltäglichen Selbstverständlichkeit geworden. Die anfängliche Euphorie über den gedanklich bereits vollzogenen Schritt ins Informationszeitalter ebbte allerdings schneller ab als erwartet, denn gerade Versprechen wie der freie Zugang zum weltweiten Wissen konnten nicht eingelöst werden. Das rapide Wachstum des WWW hätte sogar sein Erfinder, Tim Berners-Lee[1], nicht vorhersagen können und so sehen wir uns heute mit einer geschätzten Zahl von mehr mehreren Milliarden Webseiten weltweit konfrontiert. Tief verborgen in diesem Chaos befinden sich wahre 'Informationsschätze' aber leider unproportional viel mehr 'Datenmüll'.

Für mich zählte also von Anfang an auch die Internetrecherche zum studentischen Arbeiten. Dabei blieb auch mir die Erfahrung nicht erspart, trotz intensiven Zeiteinsatzes meistens doch nicht die gewünschten Informationen ausfindig machen zu können.

Die Auseinandersetzung mit dem Internet und der zugrundeliegenden Technik war einer der Gründe für mein wachsendes Interesse an Suchstrategien und -techniken, wie sie z.B. in Suchmaschinen eingesetzt werden.

Dieses Interesse beruht aber nicht alleine auf dem Umgang mit dem Internet, sondern ist auch beruflich motiviert. Im Rahmen meiner Tätigkeit als Praxisstudent der Robert Bosch GmbH in Hildesheim, wo ich seit 1999 als Administrator eines abteilungsweiten zentralen Dokumentenmanagementsystems beschäftigt bin, war die Auseinandersetzung

[1] http://www.w3.org/People/Berners-Lee/, (Zugriff: 14-May-2002)

mit Aufgaben der Informationsbeschaffung ein zentrales Thema. Hier lernte ich u.a. auch das Datenbanksystem der Firma Oracle kennen, dessen Fähigkeiten in Bezug auf das Information Retrieval in dieser Arbeit als Beispiele dienen.

Die vorliegende Arbeit ist somit als eine Ausweitung meiner persönlichen Interessen anzusehen, die durch den praktischen Umgang mit der Informationstechnologie entstanden sind. Hierzu zählt auch die objektorientierte Programmierung, speziell in der Programmiersprache Java, mit der ich im Laufe des Studiums mehrmals konfrontiert wurde.

Eine besonders interessante Aufgabe stellte die Mitarbeit an einem Projekt des Fachbereichs III der Universität Hildesheim dar, das sich mit der Entwicklung intelligenter Software-Komponenten für das Information Retrieval in Java beschäftigte. Das Projekt fand unter der Leitung von Prof. Christa Womser-Hacker statt und basiert auf dem von ihr erarbeiteten MIMOR-Modell[2], das ebenfalls in dieser Arbeit thematisiert wird. Der Grundgedanke für das Thema der Arbeit entstand durch die Auseinandersetzung mit Problemstellungen dieses Projektes. Aus diesem Grund ist eines der vorrangigen Ziele dieser Arbeit die Entwicklung einer Software-Komponente, die bei der Fortführung des Projektes hilfreich eingesetzt werden kann.

Ich würde gerne die Gelegenheit nutzen, mich bei meinen Mitbewohnern und allen Freunden und Kommilitonen zu bedanken, die mich mit ihrer seelischen Unterstützung durch diese Arbeit begleitet haben und als Korrekturleser herhalten mussten.

Außerdem möchte ich es nicht versäumen, an dieser Stelle meinen Eltern zu danken, die es immer verstanden, mir den nötigen Freiraum zu lassen, um meine eigenen Entscheidungen zu treffen.

[2] MIMOR = Mehrfachindexierung zur dynamischen Methoden-Objekt-Relationierung im Information Retrieval; cf. [WOMSER-HACKER 1996]

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Elektronische Datensammlungen und auf ihnen aufbauende Informationssysteme bilden die grundlegende technologische Ebene, die viele Bereiche des privaten, wirtschaftlichen und wissenschaftlichen Lebens berührt. Die Geschwindigkeit, mit der der Bestand an digital erhältlichen Daten wächst, wird aufgrund der zunehmenden Vernetzung, der Verfügbarkeit von Bandbreite und dem anhaltendem Einzug der Informationstechnik in alle Bereiche des täglichen Lebens weiter ansteigen. Dabei wird die Verwendung multimedialer Objekte als Informationsträger in den nächsten Jahren um ein Vielfaches zunehmen. Diese wachsende Informationsflut verdeutlicht den Bedarf an Erkenntnissen und Entwicklungen aus dem Bereich der Informationsbeschaffung, die die Lokalisierung von Daten zum Ziel hat, die für den Empfänger einen informationellen Mehrwert darstellen.

Das Interesse an Techniken, die das Auffinden interessanter Informationen unterstützen ist heute größer denn je. Dies ist besonders vor dem Hintergrund des raschen Wachstums des Internet nicht verwunderlich. Es wird von vielen als die größte öffentliche Ansammlung menschlichen Wissens angesehen. Aber vor allen Dingen in privaten Datenbeständen und Netzen, wie z.B. unternehmensinternen Intranets, existieren weiter riesige Mengen expliziten (und impliziten) Wissens, die es zu erschließen gilt. Zu den Disziplinen, die den Zugang zu und den Umgang mit diesem Wissen effektiver zu gestalten versuchen, gehören Wissensmanagement, Informationsmanagement, Information Retrieval oder auch Information Ressources Management, um nur einige zu nennen.

Die vorliegende Arbeit befasst sich mit dem funktionellen Einsatz von Information Retrieval Systemen (IR-Systemen) in einem heterogenen Umfeld. IR-Systeme setzen auf der technischen Seite die Modelle und Algorithmen des Information Retrieval um und sind für die (physikalische) Speicherung und Verwaltung von Daten verantwortlich.

Aber was ist eigentlich beim Umgang mit IR-Systemen zu beachten und welche Anforderungen werden heute an sie gestellt? Wo liegen Gemeinsamkeiten und Unterschiede verschiedener Implementierungen und wie lassen sie sich in einen größeren Kontext, z.B. als Bestandteil einer unternehmesweiten IT-Infrastruktur, integrieren?

Das Ziel dieser Arbeit ist die Entwicklung einer Software-Applikation, die es erlaubt, heterogene IR-Systeme und zusätzliche, die Suche unterstützende 'Hilfskomponenten' in einen Verteiler zu integrieren. Dies dient dem Zweck, eine simultane Suche in multiplen Datenbeständen zu ermöglichen und Vorteile aus der Bündelung von Funktionen an einem Ort zu ziehen. Dabei steht die Frage im Vordergrund, wie mit einer möglichst großen und heterogenen Menge von Protokollen, Datentypen, Anfragesprachen und Informationsobjekten umgegangen werden soll und inwieweit die Schaffung von Transparenz zur Minderung dieser Komplexität beitragen kann.

Der erste Teil der Arbeit dient zunächst der Einführung in ausgewählte Bereiche des Information Retrieval (IR). Zur Vermittlung eines Überblicks über die Bedeutung und die Anwendung des IR wird das Gebiet in den Kontext des Informationsmanagements eingeordnet. Abschließend wird das MIMOR-Modell vorgestellt, das sich mit der Verwendung multipler IR-Systeme und -Komponenten in einem lernfähigen System zur Verbesserung der Retrievalqualität beschäftigt. Hier wird das erste Mal auf das Ziel des sich im zweiten Teil anschließenden Entwicklungsprozesses eingegangen und eine Arbeitsdefinition aufgestellt.

Anschließend werden die verbreitetsten Methoden vorgestellt, die heutige IR-Systeme einsetzen, um die an sie gestellten Anforderungen zu erfüllen. Der Fokus wird im weiteren Verlauf auf die technische Umsetzung der Anforderungen gelenkt. Am Beispiel einiger Datenbankmanagementsysteme (DBMS) wird die praktische Umsetzung von Retrievalstrategien in heute erhältlicher Software näher betrachtet. Ein weiterer wichtiger Punkt des ersten Teils ist die Erzeugung von Transparenz auf Basis von Metadaten. Das Ziel des ersten Teils ist es, Schlussfolgerungen und Impulse für das Entwicklungsvorhaben zu sammeln.

Im ersten Teil der Arbeit gewonnene Erkenntnisse sollen im zweiten Teil in die Entwicklung der Software-Applikation einfließen. Dieser Teil der Arbeit repräsentiert einen Software-Entwicklungsprozess, bei dem in einem ersten Schritt neue Anforderungen identifiziert und festgehalten werden. Dieser Phase schließt sich die Modellierung und Entwicklung der Software an. Die jeweils zugrundeliegenden Gedanken und Konzepte werden präsentiert und wo nötig wird auf ihre programmiersprachliche Umsetzung eingegangen.

Das letzte Kapitel fasst noch einmal wichtige theoretische und praktische Implikationen zusammen und gibt einen Ausblick auf die Weiterentwicklung der Software.

2 Information Retrieval

Information Retrieval hat sich in den letzten Jahrzehnten stark über seine traditionelle Rolle als bloße Indexierung und Filterung von Textdokumenten in umfangreichen Kollektionen[1] hinaus weiterentwickelt. Wichtige Forschungsbereiche des IR befassen sich heute beispielsweise mit Modellierung, Kategorisierung und Indexierung, Visualisierungsmethoden, Filter- und Ranking-Algorithmen oder Benutzeroberflächen. Eine passende Definition findet sich in [BAEZA-YATES; RIBEIRO-NETO 1999, S. 7]:

> "Information Retrieval (IR) deals with the representation, storage, organization of, and access to information items."

> "Dabei gibt es grundsätzlich keine Einschränkungen in der Art der Informationen, die in Retrievalsystemen gespeichert werden können." [SALTON; MCGILL 1983, S. 1]

Die Integration heterogener IR-Systeme bedeutet u.a. die Integration von Texten und Fakten, bzw. analog die Integration von Information Retrieval und Data Retrieval Systemen (cf. Kap. 4.2). Der Unterschied zwischen beiden liegt in der Verwendung unterschiedlicher Such-Paradigmen aufgrund der Unsicherheit und Vagheit, mit der sich das IR vorrangig beschäftigt. IR-Systeme versuchen mit Hilfe des *Best-Match*-Paradigmas (cf. [WOMSER-HACKER 1996, S. 223]) auf eine vage Anfrage eine Reihe wahrscheinlich relevanter Antworten zu finden, die ein Informationsbedürfnis zu unterschiedlichen Graden erfüllen können. Die Bestimmung dieser Wahrscheinlichkeit ist die Aufgabe der Matching-Funktion (cf. [WOMSER-HACKER 1996, Abb. 2.1]), bzw. des Matching-Algorithmus. Auf diese Weise lassen sich die Ergebnisse in eine nach Relevanz geordnete

[1] Anm. d. Autors: Für interessierte Cineasten wird Terry Gilliams *Brazil* empfohlen. Dieser Film zeichnet ein sehr düsteres Bild einer überbürokratisierten Gesellschaft, deren Information Retrieval auf menschlicher Arbeitskraft aufbaut; http://german.imdb.com/Title?0088846 (Zugriff: 23-April-2002)

Reihenfolge[2] (*Ranking*) bringen, wie sie z.B. von heutigen Web-Suchmaschinen bekannt sein dürfte. Ein vages Informationsbedürfnis ist typischerweise in natürlicher Sprache formuliert:

> "Finde alle Dokumente, die die Unterdrückung von Minderjährigen durch Kinderarbeit in Dritte-Welt-Ländern behandeln!"

Demgegenüber folgt das Data-Retrieval dem *Exact-Match*-Paradigma (cf. [WOMSER-HACKER 1996, S. 223]), i.e. ein Dokument muss ein Informationsbedürfnis genau erfüllen, andernfalls würde es nicht als Antwort in Betracht gezogen. Die Matching-Funktion kennt nur zwei mögliche Ergebnisse: Ja oder Nein. Eine typische Formulierung eines solchen Informationsbedürfnisses stellt die folgende dar:

> "Wie hoch war das durchschnittliche Bruttoinlandsprodukt der BRD in den Jahren 1970 bis 1980?"

Im Gegensatz zu der vorherigen vagen Anfrage, lässt sich diese sehr einfach in einen Ausdruck der relationalen Algebra übersetzen und sich mit einer Anfragesprache wie SQL[3] ausführen. Möglich wird dies durch die Restriktivität beim Data Retrieval. Fakten sind atomare Elemente, i.e. es handelt sich um numerische Werte oder einzelne Worte, deren semantische Interpretation und Format eindeutig sind, z.B. dadurch, dass sie einem Datentyp in einem relationalen Datenbank-Schema entsprechen.

IR-Systeme dagegen handeln mit unstrukturierten Daten. Texte können beispielsweise unstrukturiert und von unterschiedlicher Länge sein. Die semantische Bedeutung einzelner Elemente (Worte, Überschriften, Paragraphen, etc.) und ihre Beziehungen zueinander sind nicht immer eindeutig bestimmbar.

Viele der erprobten und etablierten Verfahren und Modelle des IR befassen sich – aufgrund ihrer klassischen Herkunft – mit Textkollektionen. Daher werden im folgenden die Begriffe *Dokument* und *Informationsobjekt* gleichbedeutend verwendet.

[2] Der Begriff Ranking bezieht sich im IR i.d.R. auf eine Ordnung nach Relevanz. Prinzipiell kann ein Ranking aber auch anhand anderer Faktoren (Datum, alphabetisch, etc.) erfolgen

[3] SQL = Structured Query Language

2.1 Information Retrieval im Kontext des Informationsmanagements

In diesem Kapitel soll das IR zunächst in den größeren Kontext des Informationsmangements (IM) eingeordnet werden. Der Kernpunkt des IM wird oft auf das wesentliche Problem der Informationsbereitstellung reduziert, das darin besteht, "die 'richtige' Art von Information zum 'richtigen' Zeitpunkt am 'richtigen' Entscheidungsort [...]" [VOSS; GUTENSCHWAGER 2001, S. 75] zur Verfügung zu stellen.

> "Informationsmanagement ist die wirtschaftliche (effiziente) Planung, Beschaffung, Verarbeitung, Distribution und Allokation von Informationen als Ressource zur Vorbereitung und Unterstützung von Entscheidungen (Entscheidungsprozessen) sowie die Gestaltung der dazu erforderlichen Rahmenbedingungen." [VOSS; GUTENSCHWAGER 2001, S. 70]

Die Rahmenbedingungen auf technischer Ebene werden durch eine IT-Infrastruktur geschaffen, die sich, stark vereinfacht, aus Datenbeständen, Informationssystemen und Kommunikationskanälen zusammensetzt. Ein Ansatzpunkt zur Optimierung, der schon in der Planungsphase der Rahmenbedingungen berücksichtigt werden kann, ist die geeignete Auswahl der Allokations- und Distributionswege von Information als Teil der Informationslogistik.

> "Die Allokation beschreibt die Informationsverteilung (Wer bekommt, pflegt und wartet welche Informationen?); die Distribution beschreibt die Art der Informationsverteilung. [...] Grundvoraussetzung zur Gestaltung der Informationsbereitstellung – im Sinne einer effizienten Informationslogistik [...] - ist die Transparenz" [VOSS; GUTENSCHWAGER 2001, S. 76]

Bezüglich der Bereitstellung der 'richtigen' Information (s.o.) bezieht sich diese Transparenz unter anderem auf die bestehenden Datenbestände und ihre Inhalte sowie auf Vorgänge und Zusammenhänge innerhalb der IT-Infrastruktur[4].

IM ist aus heutiger Sicht sehr eng mit anderen Disziplinen wie der Informatik, der Betriebswirtschaftslehre und der Wirtschaftsinformatik verknüpft (cf. [VOSS; GUTENSCHWAGER 2001, Abb. 3.7, S. 85]). Das IR spielt besonders in der BWL eine tragende Rolle bei der Datenbeschaffung und -allokation (*Datenmanagement*; cf. [VOSS; GUTENSCHWAGER 2001, Kap. 7]), da die Informationsbeschaffung traditionell den Engpass im betrieblichen Entscheidungsprozess darstellt.

[4] Die für die unternehmensweite Transparenz notwendige Beschreibung von Objekten, ihrer Attribute, Eigenschaften und Beziehungen sorgt i.d.R ein Data-Dictionary bzw. Repository; cf. Kap. 5.3

2.1.1 Der Wert von Information

In der Volks- und Betriebswirtschaftslehre hat sich inzwischen die Verfügbarkeit und Verwertung (relevanter) Daten als Erfolgsfaktor zur "zielführenden Kombination der klassischen Produktionsfaktoren" [LEHNER 2000, S. 13] (Arbeit, Betriebsmittel, Werkstoffe) etabliert.

> "In der Volkswirtschaftslehre werden hauptsächlich die Ressourcen (Produktionsfaktoren) Arbeit, Kapital (Geld- und Sachkapital) und technisches Wissen unterschieden. Wissen wird hier als die Fähigkeit verstanden, die übrigen Faktoren möglichst günstig zu kombinieren." [VOSS; GUTENSCHWAGER 2001, S. 20]

Im Bereich der Wissensverarbeitung wird der Zusammenhang von Information und Wissen so aufgefasst, dass aus Wissen Informationen gewonnen und umgekehrt Informationen in einem anderen Kontext wieder als Wissen abgespeichert werden können (cf. [REIF 2000]). Im Kontext dieser Arbeit wird der Information daher ein weitaus größerer Wert beigemessen als ein bloßer Erfolgsfaktor. Information selber soll als derivativer Produktionsfaktor (cf. [LEHNER 2000, S. 12]) verstanden werden. Das bedeutet, dass Informationen produziert werden und wiederum als Input für weitere Produktionsprozesse dienen können.

> "Ein Ansatz ist, Weisheit mit Meta-Wissen, i.e. Wissen über die Generierung, Verfügbarmachung und Integration von Wissen, gleichzusetzen [...]. Meta-Wissen spiegelt sich dabei auch in der Fähigkeit zur Transformation von implizitem in explizites Wissen wider [...]." [VOSS; GUTENSCHWAGER 2001, S. 14]

Implizites Wissen (*tacit knowledge*) muss, damit es zur Information werden kann, zunächst geäußert, i.e. extrahiert werden (cf. [VOSS; GUTENSCHWAGER 2001, S. 10]). Eine mögliche Quelle impliziten Wissens im betrieblichen Umfeld stellt beispielsweise das spezielle Wissen einzelner Mitarbeiter[5] bezogen auf ihr Arbeitsumfeld dar. Bezieht man dieses Wissen auf ein Arbeitsumfeld, das dem IR näher liegt, z.B. eine Bibliothek, so lässt sich feststellen, dass implizites Wissen hier zur Effizienzsteigerung bei der Informationssuche eingesetzt werden kann. Genau dann nämlich, wenn der Bibliothekar dem Suchenden mit seinen Kenntnissen über den Datenbestand und seiner Erfahrung zur Seite steht.

[5] In Mitarbeiterkreisen werden die Tricks und Kniffe, die sich jemand durch Erfahrung in seinem Beruf angeeignet hat, oft als *Guru-Wissen* bezeichnet.

Eine mögliche Verwendung von Meta-Wissen im IR kann somit auf die Vermeidung von Unsicherheit bezogen werden. Durch Extraktion, Akkumulierung, Analyse und Generalisierung solch impliziten Wissens im IR ließen sich z.B. Regeln[6] ableiten. Die Bereitstellung und Nutzung dieser gewonnenen 'Weisheit' könnte die Informationssuche entscheidend verbessern. Im Rahmen dieser Arbeit wird Meta-Wissen wie folgt definiert.

> "Meta-Wissen ist die Sicherheit, sich bei jeder Art von Informationsbedürfnis, mit der 'richtigen' Frageformulierung an die 'richtigen' Informationsquellen zu wenden und sich der 'richtigen' Mittel für die Extraktion der Informationen zu bedienen."

2.1.2 Ebenen des Informationsmanagements

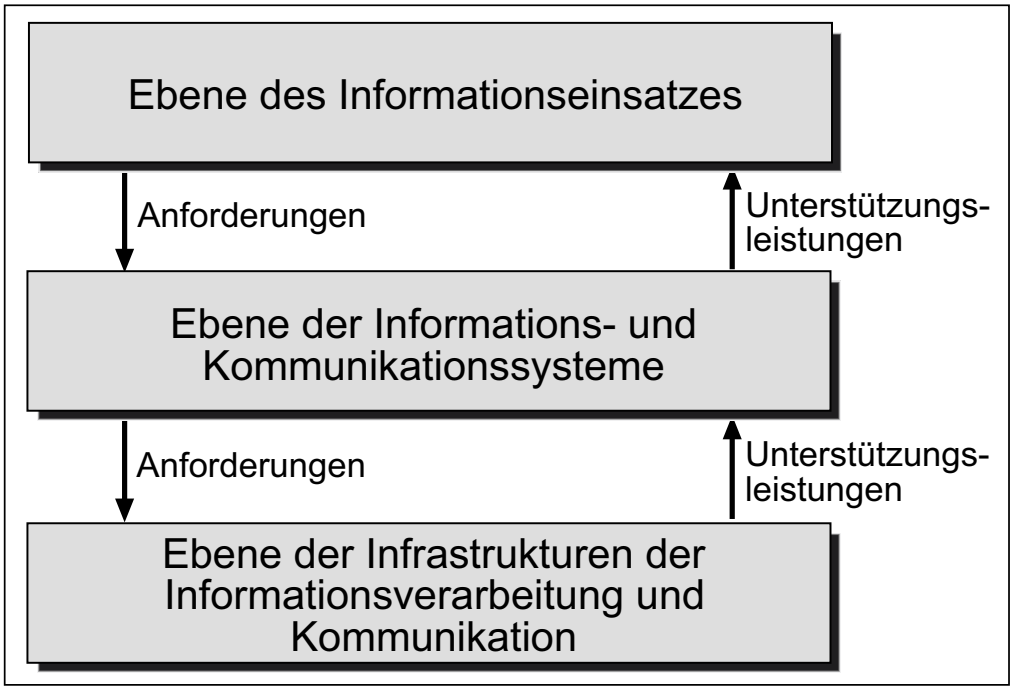

Abb. 2.1: Das Ebenen-Modell des Informationsmanagements, Quelle: [VOSS; GUTENSCHWAGER 2001, S. 72] nach [WOLLNIK 1988, S. 38]

[WOLLNIK 1988] nimmt eine Unterteilung der Aufgaben des Informationsmanagements in drei Ebenen vor, die sich durch ihre Nähe zur Informationstechnik definieren.

[6] Regelbasierte Ansätze (cf. [RUMELHART; MCCLELLAND 1986, S. 32]) werden z.B. bei Expertensystemen eingesetzt. Hierbei werden Experten eines bestimmten Gebiets (Domäne) befragt und beobachtet, um Regeln in ihrem Verhalten herauszufiltern, die dann in einem Computersystem umgesetzt werden.

Dieser Ansatz wird auch von [KRCMAR 2000] und [VOSS; GUTENSCHWAGER 2001] aufgegriffen und weiterentwickelt. Die Zusammenhänge zwischen den Ebenen sind in Abb. 2.1 dargestellt. Jede Ebene kann weiter unterteilt werden in Planung, Organisation und Kontrolle und besitzt somit strategische, taktische und operative Aufgabengebiete[7] (cf. [VOSS; GUTENSCHWAGER 2001, S. 73]). Dabei lässt sich für die Ebene des Informationseinsatzes, die in Wollniks Modell der obersten Ebene entspricht, eine Konzentration auf die Planung, also eine strategische Ausrichtung erkennen. Diese Orientierung spielt im unternehmerischen Einsatz eine wesentliche Rolle, im Kontext dieser Arbeit sollen allerdings die beiden unteren Ebenen hervorgehoben werden, da sie die grundlegenden Unterstützungsleistungen für einen effizienten Informationseinsatz erbringen und eine größere Nähe zur Informationstechnologie (IT) aufweisen.

2.1.3 Ausrichtung des Informationsmanagements

Es lässt sich keine allgemeingültige Definition für das Informationsmanagement aufstellen (cf. [VOSS; GUTENSCHWAGER 2001, S. 58ff.]). Vielmehr wird deutlich, dass sich einzelne Ansätze durch ihre Ausrichtung auf unterschiedliche Wissenschaftsbereiche und eine unterschiedliche Gewichtung des Management der Datenverarbeitung (DV) voneinander abgrenzen. [VOSS; GUTENSCHWAGER 2001, S. 62ff.] identifiziert deshalb folgende Ansätze:

1. DV-orientiertes Informationsmanagement
2. Informationsressourcenmanagement
3. Persönliches Informationsmanagement
4. Prozess-orientiertes Informationsmanagement
5. Ganzheitliches Informationsmanagement

Für diese Arbeit soll die DV-orientierte sowie die ganzheitliche, i.e. strategie-orientierte Ausrichtung eine wesentliche Rolle spielen.

> "Das Informationsmanagement muss versuchen, die Informationssystemarchitektur an die strategischen Unternehmensziele anzupassen. Hier geht es um die Optimierung der Unterstützungsfunktion der Informationstechnik

[7] [KRCMAR 2000] definiert daneben auch ebenenübergreifende Aufgaben wie das Controlling oder Personalmanagement (cf. [VOSS; GUTENSCHWAGER 2001, S. 79]). Diese Ebenen legen damit auch unterschiedliche Anforderungsprofile an Informationsmanager fest.

> durch die strategisch ausgerichtete Anpassung der betrieblichen Informationssysteme an die Unternehmensorganisation und die Bereichszielsetzungen. Krcmar (2000) stellt heraus, dass dieser Anpassungsprozess tatsächlich der traditionellen Sichtweise auf die Datenverarbeitung als reinem Dienstleister im Unternehmen entspricht.“ Dabei ist “[. . .]eine strategische Planung in diesem Bereich aufgrund der rasanten Entwicklung der IT kaum zu realisieren. Stellt man allerdings eher die Definition von Funktionalitätsanforderungen in den Vordergrund der Betrachtung und weniger die Auseinandersetzung mit am Markt verfügbarer Technik, so kann dies sicherlich differenzierter betrachtet werden.“ [VOSS; GUTENSCHWAGER 2001, S. 66f.]

Die Bezeichnung ganzheitliches Informationssystem ist also immer unter Berücksichtigung einer Strategie-Orientierung aufzufassen, die natürlich nicht allein auf Unternehmensziele begrenzt ist, sondern sich durch jede übergeordnete Strategie begründen lässt. Für die in dieser Arbeit durchgeführte Software-Entwicklung wird die Erzeugung von Meta-Wissen (cf. Kap. 2.1.1) und die Bewertung der Qualität von IR-Systemen zur Verbesserung des Retrievals als übergeordnetes Ziel aufgefasst.

> “Die Entwicklung des DV-Einsatzes in Unternehmen ist gekennzeichnet durch eine stete Ausweitung der Aufgaben, die von einem Computer übernommen bzw. mit Computerunterstützung abgewickelt werden. Das führt dazu, dass tendenziell sowohl die Zahl der eingesetzten betrieblichen Anwendungssysteme, als auch ihre Komplexität steigt. [. . .]Die möglichst reibungslose Zusammenarbeit aller Bestandteile des betrieblichen Informationssystems erfordert die Realisierung einer integrierten Datenverarbeitung.“ [MYRACH 1994, S. 1]

Für ein ganzheitliches integriertes IR-System können bis hierher folgende Schlussfolgerungen gezogen werden.

1. Die Integration heterogener IR-Systeme (Texte und Fakten) muss möglich sein.

2. Transparenz von Daten und Prozessen sollte gewährleistet werden.

3. Eine am IR ausgerichtete Strategie kann die Erzeugung von Meta-Wissen sein, i.e. Informationen können zu Wissen über die Optimierung der Informationssuche verarbeitet werden.

2.2 IR-Modelle

Ein IR-Modell stellt sich formal als Quadrupel dar (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, Kap. 2.4]), bestehend aus der logischen Sicht auf die Dokumente und der logischen Sicht auf die Informationsbedürfnisse (*Queries*) sowie einer Ranking-Funktion, die einer Repräsentation eines Dokuments und einem Query eine reelle Zahl zum Ausdruck der Ähnlichkeit zuordnet. Den nötigen Raum, in dem die Dokumentrepräsentationen und Queries sowie ihre Beziehungen zueinander modelliert und Operationen auf ihnen ausgeführt werden, stellt ein geeignetes Framework zur Verfügung. Im Falle des Vektor-Modells beispielsweise besteht das Framework aus einem n-dimensionalen Vektorraum und Standardoperationen zur Anwendung linearer Algebra auf Vektoren (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 27ff.]).

> "[...] ranking algorithms are at the core of information retrieval systems. [...] The IR model adopted determines the predictions of what is relevant and what is not (i.e., the notion of relevance implemented by the system)." [BAEZA-YATES; RIBEIRO-NETO 1999, S. 19]

In Abb. 2.2 ist eine Taxonomie der bekanntesten IR-Modelle dargestellt. Jedes Modell besitzt unterschiedliche Vor- und Nachteile in Bezug auf seinen Einsatz im IR. Klassische Boolesche Anfragen eignen sich z.B. generell eher für das Data Retrieval und das Exact-Match-Paradigma. Ein Dokument erfüllt entweder die booleschen Anforderungen oder nicht. Durch die fehlende Mehrwertigkeit der Relevanz ist es nicht möglich, ein geordnetes Ranking zu erstellen. Die Nachteile einer binären Relevanzbewertung bei der Verwendung boolescher Logik im IR soll das folgende formale Query verdeutlichen:

$$Query\ q = Term\ t_x\ AND\ Term\ t_y$$

Nach dem klassischen booleschen Modell wäre ein Dokument, das entweder den Term t_x oder den Term t_y enthält genauso irrelevant für das Query q wie ein Dokument, dass keinen der beiden Terme enthält.

Zur Behebung dieses Missstandes existieren eine Reihe von Modellen, die aus der Fuzzy-Theorie stammen. Die Fuzzy-Set-Theorie verwendet das Konzept einer graduellen Zugehörigkeit eines Elements zu einer Menge. Dieser Zugehörigkeit ist üblicherweise ein auf das Intervall [0,1] normalisierter Wert. Die Ähnlichkeit eines Query zu einem Dokument lässt sich nun aufgrund unterschiedlicher Modelle berechnen, die die Restriktivität boolescher Operatoren aufheben. Hierzu gehört das von [SALTON; MCGILL 1983] vorgestellte *Extended Boolean Model* (cf. [BAEZA-YATES; RIBEIRO-NETO 1999,

S. 38]). Eine eingehendere Behandlung der IR-Modelle findet an dieser Stelle nicht statt.

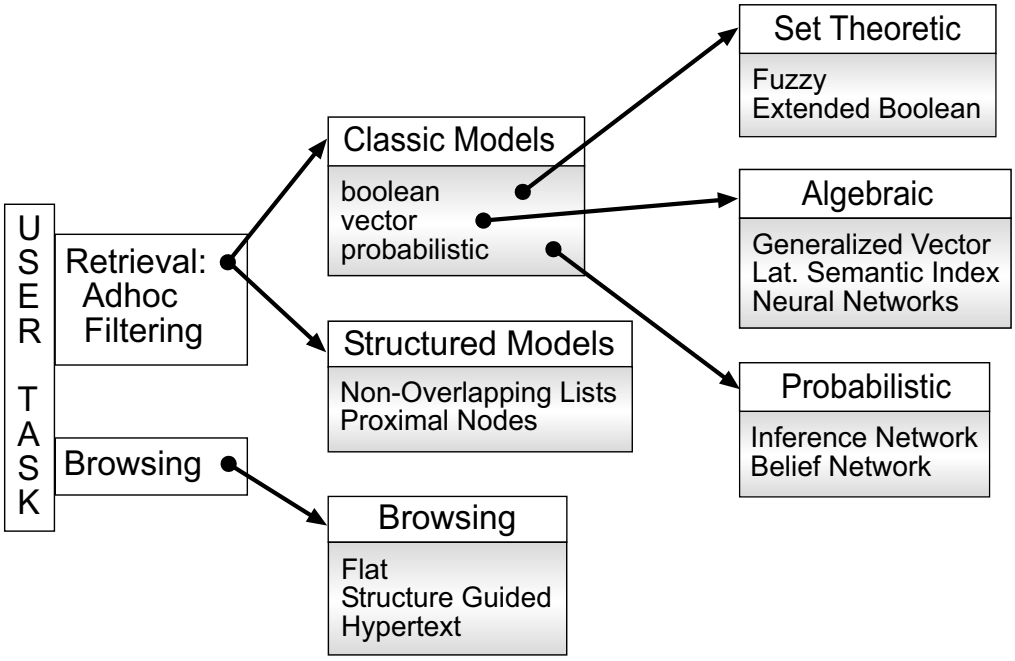

Abb. 2.2: Taxonomie von IR-Modellen,
Quelle: [BAEZA-YATES; RIBEIRO-NETO 1999, S. 21]

Einen für diese Arbeit wichtigen Aspekt stellt der *User Task* (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 4f.]), auch *Retrieval Task* genannt, dar (cf. Abb. 2.2). Während *Browsing* und *Ad-hoc-Queries* sog. Pulling-Aktionen darstellen, i.e. der Benutzer verlangt explizit nach Informationen aus einem statischen Datenbestand, existiert eine weitere Form des Retrievalauftrags, die in der Literatur als *Filtering* oder *Routing* bezeichnet wird und eine Pushing-Strategie umsetzt (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 5]). Diese Strategie beschreibt eine dynamische Veränderung der Informationsbasis, in die ständig neu eintreffende Informationen eingefügt werden (z.B. Nachrichten in einer Newsgroup) und die gegen ein statisches Informationsbedürfnis eines Benutzers verglichen werden[8].

Zur Verdeutlichung der Unterschiede zwischen diesen Techniken soll noch einmal das Beispiel der Staumeldung aus der Einleitung herangezogen werden. Es wird angenommen, das Informationsbedürfnis bestünde darin, zu erfahren, wie die Verkehrslage auf einer bestimmten Strecke ist. Auf der einen Seite könnte man sich nun der Ad-Hoc-Strategie bedienen und aktiv nach Informationen suchen oder suchen lassen. Hierfür stehen z.B. Routenplaner im WWW, Navigationssysteme oder die Telefon-Hotline des ADAC zur

[8] [KOWALSKI 1997, S. 15] bezeichnet Filtering als Selective Dissemination of Information' (Dissemination (engl.) = Verbreitung, Verteilung).

Disposition. Erfährt man von einem Stau, könnte die Browsing-Technik dazu verwendet werden, sich durch Verweise (*Links*) zu weiterführenden Informationen vorzuarbeiten, z.B. zu möglichen Umleitungsstrecken oder dem Menü an der nächsten Raststätte, an der man den Stau abwarten möchte.

Bei einer Staumeldung, die man per Radio erhält, handelt es sich allerdings um eine gefilterte Nachricht. Meldungen werden vom Radiosender gegen das statische Informationsbedürfnis an aktuellen Verkehrsmeldungen verglichen und über einen geeigneten Kommunikationskanal an die Benutzer verteilt, die ein solches Informationsbedürfnis bekunden. In diesem Fall geschieht dies einfach durch das eingeschaltete Autoradio und die Wahl eines Senders mit Verkehrsfunk. Das Filtering von Informationen wird in dieser Arbeit nicht weiter verfolgt. Es erfolgt eine Konzentration auf Ad-hoc-Queries, da sie zu den am häufigsten verwendeten Anfrage-Techniken gehören. Ergänzend werden auch Kombinationsmöglichkeiten mit Browsing-Techniken behandelt.

Das nächste Kapitel befasst sich nun weiter mit dem Retrievalprozess, der die Schritte zur Durchführung des User Task beschreibt.

2.3 Der Retrievalprozess

Der Retrievalprozess erscheint aus der Sicht des Benutzers eines Retrievalsystems als eine Art *Black Box*, in die, einfach gesagt, eine Anfrage hineingegeben wird und aus der Ergebnisse herauskommen. Er wird von den internen Abläufen weitestgehend abgeschirmt und lediglich mit dem Anfangs- und Endpunkt des Prozesses konfrontiert.

Ziel dieses Kapitels ist es nun, das ’Innenleben’ dieser Black Box genauer zu betrachten. Man kann sich sicherlich vorstellen, dass abhängig von IR-System und Datenbestand der Grad der Komplexität des Prozesses stark variiert. Bevor die Behandlung von Subprozessen und Komponenten, aus denen sich der Retrievalprozess zusammensetzt, vorgenommen werden kann, sollen kurz einige Voraussetzungen für die Durchführung des Prozesses angesprochen werden.

Bevor der Retrievalprozess überhaupt eingeleitet werden kann, gilt es zunächst, die Dokumentkollektion vorzubereiten, indem eine logische Sicht auf die Dokumente aufgebaut wird. Diese Sicht besteht historisch bedingt meist aus einer Menge von Indextermini, die entweder maschinell oder intellektuell (von Experten) aus dem Originaldokument extrahiert werden. Ein Index ist einfach gesagt eine optimierte Struktur, die die schnellere Suche in großen Datenbeständen ermöglicht. Aufgrund der

Rechenleistung moderner Computer ist es inzwischen aber auch möglich, die gesamte Menge der Wörter eines (Text-) Dokuments als seine logische Repräsentation zu verwenden. Man spricht in diesem Fall von einer Volltextrepräsentation der Dokumente (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, Kap. 1.2.2]). Zur Reduzierung der Komplexität der logischen Sicht werden zusätzlich verschiedene (Text-) Operationen auf dem Originaldokument ausgeführt, die die logische Sicht von einem Volltext schrittweise in eine abstraktere Repräsentation überführen. Dazu gehören Verfahren wie die Stopp-Wort-Eliminierung und Stemming, die in Kap. 3.1.1 noch einmal genauer beschrieben werden.

Die Komplexität der logischen Sicht ist zusätzlich davon abhängig, welche Anfragesprachen (cf. Kap. QueryLanguages) verwendet werden sollen. Die Unterscheidung in *Content* und *Structure* verdeutlicht die unterschiedliche Herangehensweise. Zur Einbeziehung der Struktur eines Dokuments in die Anfrage muss die logische Sicht zusätzliche Angaben zur Position von Indextermen in strukturellen Elementen wie Kapitel, Unterkapitel, Abschnitte, etc.[9]. enthalten (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 192]). Eine inhaltsbezogene Anfrage missachtet jegliche Struktur und bezieht die Anfrage auf den gesamten Text oder auf einzelne Teile, z.B. Sätze oder Wortgruppen.

Der generelle Ablauf des Retrievalprozesses wird bestimmt durch den Retrieval Task (cf. Kap. 2.2), der durch den Benutzer vorgegeben wird. Eine Unterscheidung dieses Auftrags richtet sich an der zugrundeliegenden Datenstruktur aus, in der nach Informationen gesucht werden soll. Klassische IR-Systeme ermöglichen i.d.R das Information oder Data Retrieval (cf. Kap. 2) mit Hilfe von Ad-hoc-Anfragen, während z.B. Hypertext-Systeme das 'Stöbern' (Browsing) in Dokumenten unterstützen. Der fundamentale Unterschied zwischen diesen Typen begründet sich im Informationsbedürfnis des Benutzers, das beim Browsing zu Beginn noch unbekannt oder sehr vage sein kann und erst durch die Interaktion mit dem System, i.e. während der 'Reise' durch verschiedene Dokumente, eine konkrete Form annimmt.

Der Retrievalprozess unterteilt sich in eine Reihe von Subprozessen, auf die in Kap. 3 noch einmal genauer eingegangen wird.

> "Whenever a user wants to retrieve a set of documents, he first builds up a conceptualization of what he's looking for. Such coneptualization is what we call his information need." [BAEZA-YATES; RIBEIRO-NETO 1999, S. 172]

[9] Die Struktur muss nicht auf das Dokument beschränkt sein, sondern kann auch, wie beim Hypertext (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 108]), dokumentübergreifend aufgefasst werden.

Der Prozess wird also durch die Formulierung eines Informationsbedürfnisses eingeleitet und endet mit der Übergabe eines Rankings. Abbildung 2.3 veranschaulicht diesen Prozess anhand einer typischen IR-System-Architektur. Die Buchstaben A bis E in der Abbildung sind als Referenzpunkte eingefügt worden.

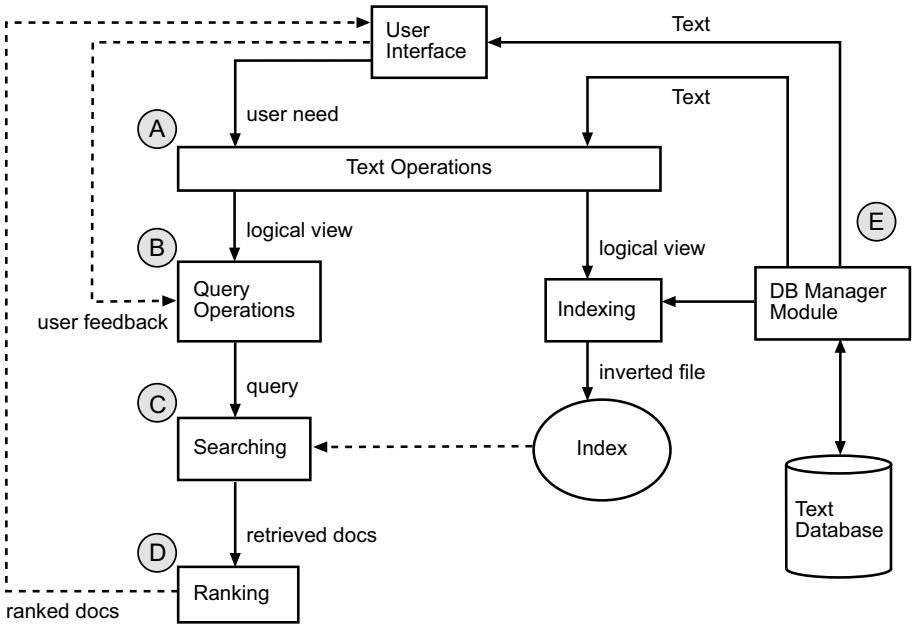

Abb. 2.3: Die Stationen des Retrievalprozesses,
Quelle: [BAEZA-YATES; RIBEIRO-NETO 1999, S. 10]

Der erste Schritt im Retrievalprozess ist die Transformation des Informationsbedürfnisses in die Anfragesprache (*Query Language*) des IR-Systems (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, Kap. 4]). Diese Umwandlung, die die Erzeugung einer systeminternen Repräsentation des Informationsbedürfnisses (*Query*) zum Ziel hat, gliedert sich in zwei Subprozesse. In einem ersten Schritt (A) werden auf die Formulierung des Informationsbedürfnisses dieselben (Text-) Operationen angewandt wie bei der Erstellung der logischen Sicht auf die Dokumente. Durch diese Normalisierung ist es wesentlich einfacher, die logischen Repräsentationen des Dokuments und des Informationsbedürfnisses miteinander zu vergleichen (Matching). In einem optionalen zweiten Schritt (B) können weitere Operationen (*Query Operations*) angewandt werden, die automatisch oder in Interaktion mit dem Benutzer durchgeführt werden (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, Kap. 5]). Sie haben meist eine exaktere Formulierung des Informationsbedürfnisses zum Ziel, zu der der Benutzer alleine normalerweise nicht im Stande ist. Die meisten Benutzer eines IR-Systems verfügen nämlich i.d.R., wie eingangs des Kapitels erwähnt, über kein Hintergrundwissen in Bezug auf Text- und Query-Operationen, was ein inadäquat formuliertes Informationsbedürfnis

zur Folge hat. "Therefore, it is not surprising, to observe, that poorly formulated queries lead to poor retrieval [...]" [BAEZA-YATES; RIBEIRO-NETO 1999, S. 10]. Eine große Mitschuld an dieser Misere tragen übrigens auch viele User Interfaces (*UIs*), die den Benutzer dazu zwingen, sein Informationsbedürfnis in einer schwer formulierbaren, aber 'systemfreundlichen' Form zu spezifizieren. Das 'Stochern' nach Informationen mit Hilfe einzelner Suchworte in Web-Suchmaschinen ist hier das beste Beispiel.

Der nächste Schritt (C) des Retrievalprozesses besteht in der Ausführung der Anfrage, indem für jedes Query-Dokument-Paar mit Hilfe der Ranking-Funktion (cf. Kap. 2.2) ein Relevanzwert ermittelt wird. Die kalkulierte Relevanz repräsentiert die (wahrscheinliche) Zufriedenstellung des Informationsbedürfnisses des Benutzer durch ein bestimmtes Dokument.

Mit Hilfe der Relevanz ist es in einem nächsten Schritt möglich, die 'Treffer' in einem Ranking anzuordnen, das als Antwort an den Benutzer zurückgegeben wird. Üblicherweise enthält diese Antwort lediglich Auszüge aus (Titel, Abstract, etc.) und Verweise auf die entsprechenden Dokumente, da dem Benutzer eine minimale Informationsmenge meist genügt, um zu entscheiden, ob das Dokument relevant ist oder nicht. Durch die Selektion bestimmter Bereiche eines Dokuments für die Anzeige im Ranking (*Zoning*) lassen sich mehr Treffer auf einer Seite darstellen, was die Übersichtlichkeit erhöht (cf. [KOWALSKI 1997, S. 37]). Für den Fall, dass der Zugriff auf das vollständige Dokument gewünscht wird, kann dieses nachträglich über den Verweis aus der Datenbank extrahiert werden (E)[10].

Nach Erhalt des Rankings kann der Benutzer entscheiden, ob er eine erneute Suche anhand bestimmter Dokumente des Rankings durchführt, die er als eindeutig relevant beurteilt. Diese Art der Suche bezeichnet man auch als *User Feedback* oder *Relevance Feedback* (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 118], da die Grundlage der erneuten Suche die vom Benutzer identifizierten Dokumente sind. Sie ist besonders in IR-Systemen, die ein Vektor-Modell umsetzen sehr erfolgversprechend, da sich die Bestimmung der Ähnlichkeit (*Similarity*) von Dokumenten z.B. durch Korrelation ihrer Vektoren (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 27] und [KOWALSKI 1997, S. 153]) als relativ unkompliziert darstellt. Generell kann der Benutzer aber auch eine erneute Suche mit einer überarbeiteten Formulierung seines Informationsbedürfnisses in Angriff nehmen. Begriffe, die der Verfeinerung der Formulierung dienen, weil sie dem Benutzer vielleicht nicht eingefallen sind, kann dieser z.B. aus den Dokumenten des im ersten Durchgang gelieferten Rankings entnehmen. Ein Retrievalprozess, der aus mehreren solcher Durchgänge (Cycles) besteht und der schrittweisen Annäherung an ein verbessertes Retrievalergebnis dient wird allgemein auch als iterative Suche bezeichnet

[10] In Abb. 2.3 wird diese Funktion durch den Datenbankmanager bereitgestellt.

(cf. [KOWALSKI 1997, S. 19]).

Folgende wichtige Punkte in Bezug auf den Retrievalprozess sind festzuhalten:

1. Der Prozess besteht aus mindestens einem, möglicherweise aber mehreren Durchläufen.
2. Jeder Durchlauf setzt sich aus einer Abfolge von Operationen zusammen, die unterschiedlichen Phasen des Prozesses zugeordnet werden können. Als Hauptphasen können festgehalten werden:
   (A) Entgegennahme des Informationsbedürfnisses
   (B) Query-Operationen (Query-Erzeugung, -Expandierung sowie User Feedback)
   (C) Matching
   (D) Ergebnis-Operationen (Erstellung des Ranking, Relevanznormalisierung)
   (E) Zugriff auf Original-Dokumente
3. Durch eine zeitliche Orientierung der Phasen am Matching lassen sich Pre- und Post-Matching-Phasen unterscheiden. Dies hebt noch einmal die Kernfunktion des Ranking-Algorithmus (cf. Kap. 2.2) innerhalb eines IR-Systems hervor.

Eine Untersuchung des Rankings als Endergebnis des Retrievalprozesses lässt Rückschlüsse auf die Qualität des IR-Systems zu. Das folgende Kapitel soll deshalb veranschaulichen, auf welche Weise die Performanz von IR-Systemen bewertet werden kann.

2.4 Information Retrieval Evaluierung

Die Bewertung eines IR-Systems kann anhand unterschiedlichster Faktoren erfolgen. Kriterien können Kosten- und Zeit- sowie Software-ergonomische Faktoren sein, z.B. das Verhalten bei Fehlern oder die Benutzerfreundlichkeit. Eine solche Evaluierung wird als *Performance Evaluation* bezeichnet (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 73]), die allerdings wenig aussagt über die Qualität eines IR-Systems gemessen an der Güte der Ergebnisse. Hiermit befasst sich die *Retrieval Performance Evaluation*.

Die Grundlage für diese Form der Evaluierung bildet das für eine Anfrage zurückgelieferte Ranking. Unter Zuhilfenahme einer Referenzkollektion können Maßzahlen ermittelt werden, die Aufschluss über die Qualität des Rankings geben. Die beiden meist verwendeten Maße in diesem Zusammenhang sind *Recall* und *Precision*, auf die im folgenden eingegangen wird. Referenzkollektionen werden im darauffolgenden Kap. 2.4.2 behandelt.

2.4.1 Kennzahlen

Recall

Recall beschreibt den Anteil der gefundenen, relevanten Dokumente $|R_a|$ an der Gesamtzahl der relevanten Dokumente in der Kollektion $|R|$.

$$Recall = \frac{|R_a|}{|R|}$$

Precision

Precision beschreibt den Anteil der gefundenen, relevanten Dokumente $|R_a|$ an der Anzahl der gefundenen Dokumente in der Ergebnisliste $|D|$.

$$Precision = \frac{|R_a|}{|D|}$$

Fallout

Als Fallout bezeichnet man den Anteil der gefundenen, nicht relevanten Dokumente $|!R_a|$ an der Gesamtzahl der nicht relevanten Dokumente in der Kollektion $|!R|$.

$$Fallout = \frac{|!R_a|}{|!R|}$$

Die Kennzahlen Recall und Precision beschreiben die Performanz eines IR-Systems dahingehend, dass sie beim Durchlaufen des Rankings für jede Momentaufnahme ein Gütemaß für den bisherigen Erfolg widerspiegeln. Einfacher umschrieben bedeutet das, dass wenn im Ranking nur ein Dokument enthalten ist und dieses auch noch relevant ist, man 100% relevante Treffer im Ranking und damit 100% Precision erreicht hat. Wird festgestellt, dass dieses Dokument das einzige relevante für diese Anfrage in der gesamten Kollektion ist, so hat man 100% der relevanten Dokumente der Kollektion bereits gefunden und 100% Recall erreicht. Ist allerdings ein zweiter, nicht relevanter Treffer im Ranking, so ändert sich durch die Einbeziehung dieses Treffers die Precision auf 50% während der Recall bei 100% bleibt, da ja bereits alle relevanten Dokumente gefunden sind.

Beim Durchlaufen des Rankings lassen sich die Recall- und Precision-Werte als Graphen darstellen, sei es im Verhältnis zueinander (Recall/Precision-Graph) oder zu anderen Größen der Kollektion wie dem Fallout (Recall/Fallout-Graph).

Da sich für jede durchgeführte Anfrage Recall-, Precision- und Fallout-Werte berechnen lassen, werden diese i.d.R. gemittelt und zueinander in Beziehung gesetzt. Die Abbildungen, die man auf diese Art und Weise erhält dienen dann dem Vergleich der Performanz von Retrieval-Strategien. Zu den typischen Abbildungen zählen u.a. folgende Graphen (cf. [WOMSER-HACKER 1996, S. 197], [KOWALSKI 1997, S.236] und

[BAEZA-YATES; RIBEIRO-NETO 1999, S.78f.]):

- Recall vs. Precision
- Recall vs. Fallout
- Average Precision
- Interpolierte Precision vs. Recall

Schließlich spielen auch noch andere Faktoren wie die Komplexität des Retrievalprozesses ein maßgebliche Rolle bei der Interpretation dieser Zahlen. Es wirkt sich i.d.R. stark auf die Ergebnisse aus, ob der Retrieval Task in einem Durchgang oder iterativ in Interaktion mit dem Benutzer durchgeführt wurde. Beim Vergleich von Retrievalstrategien, die User Feedback miteinbeziehen, verführen Recall und Precision zu Fehlinterpretationen, wenn sie die vom Benutzer in einem vorherigen Durchgang bereits als relevant markierten Dokumente in den Vergleich miteinbeziehen. Um die effektive Performanz-Steigerung eines modifizierten Query zu ermitteln, sollte das Ranking mit der Restkollektion (*Residual Collection*) verglichen werden, die sich ergibt, wenn man die bereits als relevant markierten Dokumente von der Gesamtkollektion abzieht. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 123] schreibt dazu weiter:

> "[...] our main purpose is to compare the performance of distinct relevance feedback strategies (and not to compare the performance before and after feedback). Thus, as a basic rule of thumb, any experimentation involving relevance feedback should always evaluate recall-precision figures relative to the residual collection."

Methoden der Anfrageoptimierung haben i.d.R. einen starken Einfluss auf Recall und Precision. Die Anfrageerweiterung durch Synonyme aus einem Thesaurus beispielsweise führt meist zu einem verbesserten Recall bei gleichzeitiger Verringerung der Precision. Dies kommt zustande, da durch die Einbeziehung zusätzlicher Suchworte zwar tendenziell mehr relevante Dokumente gefunden werden, ihre Anteil an der Treffermenge aber abnimmt, da auch immer mehr irrelevante Dokumente zurückgeliefert werden.

2.4.2 Referenzkollektionen

Referenzkollektionen [BAEZA-YATES; RIBEIRO-NETO 1999, S. 84ff.] helfen bei der Analyse der Performanz und der Qualität eines IR-Systems bemessen an den Ergebnissen für festgelegte Anfragen (*Topics*). Die Parameter zur Berechnung von Recall und Precision werden im Rahmen der Kollektionen veröffentlicht.

Man sollte sich bei der Interpretation dieser Ergebnisse allerdings auch vor Augen führen, dass ein gewisses Maß an Subjektivität nicht vermeidbar ist, da über die Relevanz von Dokumenten immer noch einzelne Individuen entscheiden müssen.

TREC

Die wohl größte Initiative, die eine Art Wettbewerb der Teilnehmer und ihrer Implementierungen von IR-Systemen untereinander erlaubt, ist die ***T**ext **RE**trieval **C**onference*, die vom NIST[11] geleitet wird. Die derzeit sieben Dokumentkollektionen stammen vollständig aus amerikanischen Quellen, unter denen Printmedien einen großen Anteil besitzen. Sie umfassen jeweils zwischen wenigen tausend bis über 200.000 Dokumente, die – ebenso wie die Topics und Resultate – durch SGML[12]-Auszeichnungen strukturiert sind. Die Relevanzbestimmung erfolgt bei TREC a posteriori. Das bedeutet, dass die Top-100 Ergebnisse pro Topic der teilnehmenden IR-Systeme zur Bewertung in einem Pool (*Pooling Method*) gesammelt werden (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 89]). Erst nach dem Retrieval wird intellektuell über die Relevanz jedes Dokuments aus diesem Pool entschieden.

Cystic Fibrosis

Diese Kollektion[13] umfasst derzeit 1239 Dokumente aus der MEDLINE-Datenbank der National Library of Medicine, die mit dem Stichwort 'Cystic Fibrosis' indexiert sind. Die Dokumente werden von vier Experten mit insgesamt vier Relevanzwerten versehen. Die Besonderheit dieser Kollektion ist, dass sich die ca. 100 Anfragen (Information Requests) überschneiden, so dass besonders IR-Systeme, die die Resultate vergangener Anfragen in den Retrievalprozess einfließen lassen davon profitieren können (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 94f.]).

Betrachtungsgegenstand der vorgestellten Evaluierungsstudien ist die Gesamtleistung eines IR-Systems. Für eine Untersuchung, welchen Anteil einzelne Verfahren

[11] National Institute of Standards and Technology; http://www.nist.gov/ (Zugriff: 27-Juli-2002)
[12] Standardized General Markup Language, cf. [GOLDFARB 2002, S. 22f.]
[13] Die Cystic-Fibrosis-Kollektion ist auf der beiliegenden CD enthalten sowie online erhältlich, z.B. unter der URL http://www.sims.berkeley.edu/ hearst/irbook/cfc/cfc.zip (Zugriff: 3-August-2002).

und Komponenten an der Qualität des Ergebnisses haben, war bisher kein Raum vorhanden bzw. es existieren keine geeigneten Hilfsmittel. [WOMSER-HACKER 1996] macht darauf aufmerksam, dass es zwar gesicherte Erkenntnisse über die Effektivität verschiedener IR-Verfahren gibt, diese allerdings unter kontrollierten Bedingungen erbracht worden sind, i.e. "in festgelegten situativen Kontexten unter stabil gehaltenen Parametern werden bestimmte Variablen kontrolliert modifiziert und die Auswirkungen auf die sog. abhängigen Variablen beobachtet. Das Problem ist nun, diese Ergebnisse zu generalisieren und auf andere Umgebungen zu übertragen. Dabei ist es notwendig, die Wirkungsweise einzelner Systemkomponenten zu isolieren, um auch Erkenntnisse über ihre Wechselwirkungen herausfinden zu können" [WOMSER-HACKER 1996, S. 268].

2.5 Das MIMOR-Modell

Eines der Hauptanliegen des MIMOR-Modells[14] ist es, eine differenziertere Betrachtung der Faktoren zu ermöglichen, die zu einem bestimmten Retrievalergebnis geführt haben. Zu diesen Faktoren gehören auf der einen Seite Objekte wie die Anfrage und das Informationsobjekt und auf der anderen Seite Verfahren, die auf diese Objekte angewandt werden. Die Kombination von Verfahren und Objekten findet im Retrievalprozess statt.

> "MIMOR ist als Testmodul konzipiert, das in den Retrievalprozess eingeschoben wird und die Korrelation zwischen Deskribierungsmethoden und Objekteigenschaften ermittelt." [WOMSER-HACKER 1996, S. 284]

Das MIMOR-Modell stützt sich dazu auf der einen Seite auf die Erfassung der Objekteigenschaften, zu denen z.B. Dokumentlänge, Kollektionsgröße, Dokumenttyp, etc. gehören, und auf das Konzept der *Mehrfachdeskribierung* cf. [WOMSER-HACKER 1996, S. 286f.]) auf der anderen Seite.

Durch die Mehrfachdeskribierung sollen Synergien ausgenutzt werden, die durch die Deskribierungsvielfalt und die Kombination verschiedener Deskribierungsverfahren in einer parallelen Suche entstehen. Dieser Effekt liegt darin begründet, dass die Anwendung verschiedener Verfahren zu unterschiedlichen Ergebnismengen führt, bei denen eine geringe Überschneidung der relevanten Dokumente vorliegt (cf. [WOMSER-HACKER 1996, S. 297]). Durch eine parallele Nutzung der Verfahren und die anschließende Fusion der Ergebnismengen kann ein qualitativ besseres Resultat erzielt werden.

[14] MIMOR = Mehrfachindexierung zur dynamischen Methoden-Objekt-Relationierung im IR

Um nun die am besten geeignete Faktorenkombination für einen konkreten Retrievalprozess zu ermitteln, werden die Verfahren und die Objekteigenschaften in einer lernenden Komponente relationiert. Durch Relevance Feedback soll diese Komponente lernen, welche Faktoren zu guten und welche zu schlechten Ergebnissen beigetragen haben. Dadurch kann sie die Gewichtung der Relationen dynamisch anpassen. Auf diese Weise soll die Lern-Komponente 'Erfahrungen' sammeln, die wieder in den Retrievalprozess einfließen können, indem nämlich – basierend auf den Erfahrungen – versucht wird, in konkreten Anwendugsfällen für ein Informationsbedürfnis die besten Verfahren auszuwählen.

Gerade dieser letzte Aspekt verdeutlicht noch einmal die Bedeutung und den potentiellen Nutzen eines lernenden Modells und seine Nähe zur menschlichen Informationsverarbeitung. Dies ist mit ein Grund, weshalb für MIMOR konnektionistische Lern-Modelle favorisiert werden (cf. [WOMSER-HACKER 1996, S. 69ff., 286]).

Bezugnehmend auf den Retrievalprozess kann MIMOR sowohl auf der Seite der Pre-Matching-, als auch auf der Seite der Post-Matching-Prozesse eingebunden werden.

Konkret bedeutet das, dass der aktuelle 'Lern-Zustand' des MIMOR-Modells in Form der gewichteten Relationen als Grundlage genommen wird, um die Daten des Retrievalprozesses zu modifizieren und aufzuwerten oder umgekehrt die Daten zur Anpassung des Lern-Zustandes zu verwenden. Die Anpassung der Gewichte erfolgt während der Pre-Matching-Phase, indem über Relevance Feedback (cf. [WOMSER-HACKER 1996, S. 309f.]) eine Bewertung des Retrievalergebnisses erfolgt und damit eine Bewertung der Faktoren, die zu dem Ergebnis beigetragen haben, möglich ist. Die Einflussnahme der Gewichte, z.B. auf die Fusionierung einzelner Rankings zu einem Gesamtranking, ist dagegen Teil der Post-Matching-Phase.

2.5.1 Metasuche in heterogenen IR-Systemen

Vielen Internetnutzern werden Meta-Suchmaschinen im Zusammenhang mit der Suche nach Webseiten im World Wide Web (WWW) ein Begriff sein. Eine Meta-Suchmaschine dient dort der Verteilung von Anfragen an eine Reihe unterschiedlicher Suchmaschinen und der Zusammenführung der Ergebnisse in einer verschmolzenen Trefferliste[15]. Folgendes Zitat dient der Definition von Metasuche:

> "a search technique common on the World Wide Web where a single point of entry is provided to multiple heterogenous back-end search engines. A meta search system sends a users's query to the back-end search engines,

[15] Beispiele für Meta-Suchmaschinen im WWW sind WebCrawler (http://www.webcrawler.com) oder MetaGer (http://www.metager.de), (Zugriffe: 23-Juli-2002)

> combines the results, and returns a single, unified hit-list to the user.“ [BAEZA-YATES; RIBEIRO-NETO 1999, S. 103]

Diese grundlegende Technik findet im Zusammenhang mit der Mehrfachdeskribierung auch Anwendung im MIMOR-Modell, wo eine einzelne Anfrage mit einer Reihe heterogener IR-Systeme und Verfahren verknüpft wird und die Ergebnislisten zu einer einzigen fusioniert werden.

Im Vorwort wurde bereits erwähnt, dass das grundsätzliche Konzept, das in dieser Arbeit thematisiert wird, bereits während der Teilnahme an einem Projektseminar im Wintersemester 2000/2001 entstand, das die Entwicklung von Komponenten für das MIMOR-Modell in der Programmiersprache Java zum Ziel hatte.

Obwohl die ausführliche Vorstellung und Ausarbeitung dieses Konzeptes erst in späteren Kapiteln erfolgt, dient die vorgezogene Einführung an dieser Stelle dazu, die grundlegende Idee erst einmal vorzustellen. Auf diese Weise wird es möglich, an den Stellen, an denen es wichtig erscheint, auf das Konzept zu verweisen und mögliche Implikationen zu verdeutlichen.

2.5.2 Der Retrieval Component Integrator

In besagtem MIMOR-Projekt war folgende Problemstellung gegeben. Es sollte eine Software-Komponente entwickelt werden, an die sich eine beliebige Anzahl heterogener IR-Systeme über Schnittstellen-Objekte anschließen lässt. Eingehende Anfragen sollten an die IR-Systeme verteilt und die zurückgelieferten Ergebnislisten in eine einheitliche Formatierung überführt werden, so dass sie von einer weiteren Komponente des Systems zu einer Gesamtliste verschmolzen werden können. Der Charakter dieser Komponente entspricht der Definition der Metasuche (s.o.) und kommt in seiner Funktion der eines Verteilerkastens gleich, durch den ein Strom von Anfragen und Antworten zwischen IR-Systemen und Benutzeranwendungen fließt.

Im folgenden soll der Begriff *Retrieval Component Integrator* oder kurz *RECOIN* repräsentativ für die Applikation verwendet werden, deren Entwicklung das erklärte Ziel dieser Arbeit ist. Die Wahl dieses Begriffes soll bewusst eine Nähe oder Verwandtschaft zum MIMOR-Modell und dessen sich dynamisch anpassender Lern-Komponente ausdrücken (recoin (engl.) = neu prägen, umprägen).

Die folgenden Anforderungen sollen grundlegende Ansprüche skizzieren und als Arbeitsdefinition dienen.

1. Einbindung multipler Schnittstellen (Adapter) zu IR-Systemen

2. Entgegennahme und Verteilung von Anfragen auf multiple Adapter
3. Rücknahme und ggf. Normalisierung der Rankings
4. Fusion der Rankings zu einem Gesamtranking

Im weiteren Verlauf der Arbeit werden zusätzliche Anforderungen deutlich werden, so dass in Kap. 6 eine Neuorientierung des Anforderungsprofils vorgenommen werden kann.

3 Stationen des Retrievalprozesses

Die grundlegenden Stationen des Retrievalprozesses wurden wurden in Kap. 2.3 bereits beschrieben und auf eine mögliche zeitliche Ausrichtung am Matching wurde hingewiesen. Darauf aufbauend werden im folgenden die Stationen des Retrievalprozesses, die zwischen der Initialisierung eines *Retrieval Cycle* (cf. Kap. 2.3) und dem Matching stattfinden als Pre-Query-Operationen bezeichnet. Analog werden alle Operationen nach dem Matching als Post-Result-Operationen angesehen.

In diesem Kapitel sollen diese Operationen im einzelnen noch einmal vorgestellt werden. Die Auflistung und Kurzbeschreibung einiger dieser Operationen in den folgenden Abschnitten soll dabei lediglich einen Eindruck von der Heterogenität der Verfahren vermitteln und stellt keinen Anspruch an Profundität oder Vollständigkeit. Die entscheidende Ergänzung dieser Auflistungen ist in der Identifikation zusätzlicher Operationen zu sehen, die sich dadurch ergeben, dass der Retrievalprozess in einer verteilten Umgebung abläuft (cf. Kap. 2.5.1).

3.1 Pre-Query-Operationen

Zu den Pre-Query-Operationen zählen *Text Operations*, *Query Language* und *Query Operations* (cf. Abb. 2.3).

3.1.1 Text Operations

In Kap. 2.3 wurde bereits erläutert, dass zur Transformation in ein Query verschiedene (Text-) Operationen, die z.B. der Normalisierung dienen, angewandt werden. Zwei der gebräuchlichsten Operationen sind:

- Stemming
- Stop-Wort-Eliminierung

Beim Stemming werden einzelne Terme auf ihre Wortstämme reduziert, um das Auffinden morphologischer Varianten zu ermöglichen. Diese Form der Normalisierung wird verwendet, um einen höheren Recall (cf. Kap. 2.4.1) zu erzielen (cf. [KOWALSKI 1997, S. 66]).

Die Eliminierung gebräuchlicher Worte, die den Großteil eines Textes ausmachen, aber keinen Einfluss auf die Retrievalqualität besitzen, bezeichnet man als Stop-Wort-Eliminierung (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 167]). Zu den Stop-Wörtern gehören z.B. Artikel, Präpositionen etc. Konventionelle Stop-Wort-Listen umfassen ca. 100 bis 600 Worte (cf. [WOMSER-HACKER 1996, S. 102]).

3.1.2 Query Language

Die Query Language beschreibt die Art der Anfrage, die an IR-Systeme gerichtet werden kann. Der mögliche Einsatz der Query Language hängt z.T. vom verwendeten IR-Modell ab. (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 99]). Query Languages werden in [BAEZA-YATES; RIBEIRO-NETO 1999, Kap. 4] behandelt und in [KOWALSKI 1997, Kap. 2], wo sie unter *Search Capabilities* eingeordnet sind. Da sich in einigen Fällen die in diesen Quellen verwendeten Namen unterscheiden, sie aber dieselbe Technik beschreiben, werden in [KOWALSKI 1997] verwendete abweichende Benennungen in Klammern vermerkt.

Single-Word- / Keyword-Queries

Keyword-Queries sind der elementarste Querytyp, der von praktisch allen IR-Systemen unterstützt wird. Die Anfrage besteht aus einem einzelnen Suchwort. “Keyword-based queries are popular because they are intuitive, easy to express, and allow for fast ranking.“ [BAEZA-YATES; RIBEIRO-NETO 1999, S. 100]

Boolean Queries ([KOWALSKI 1997]: Boolean Logic)

Elemente (*Tokens*[1]) der Suchanfrage werden mit Hilfe Boolescher Operatoren, typischerweise AND, OR und NOT, zu einem Gesamtausdruck verbunden. Die Priorität der Operatoren gehorcht entweder einer Voreinstellung oder kann durch die Verwendung von Klammern beeinflusst werden. Da Boolesche Ausdrücke entweder erfüllt sind oder nicht,

[1] Der Begriff ’Token’ bedeutet wörtlich übersetzt Zeichen und kennzeichnet in diesem Zusammenhang (Wort-) Einheiten bestehend aus mindestens einem Buchstaben, die i.d.R. durch Leerzeichen voneinander getrennt sind.

kann beim klassischen Booleschen Retrieval kein Ergebnis-Ranking vorgenommen werden (cf. Kap. 2.2).

Fuzzy Boolean Queries ([KOWALSKI 1997]: Weighted Search)

Fuzzy Boolean Queries stellen eine Erweiterung Boolescher Anfragen dar, die ein Ranking der Ergebnisse ermöglicht, indem nicht alle Booleschen Restriktionen erfüllt werden müssen. Je mehr allerdings erfüllt sind, desto höher wird ein Treffer bewertet (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 103]). Die Tokens einer Suchanfrage werden mit Gewichten versehen, um den Grad ihrer Wichtigkeit zu kennzeichnen. Als Grundlage dienen i.d.R. Boolean oder Natural Language Queries (cf. [KOWALSKI 1997, S. 169f.]).

Structural Queries

Mit Hilfe von Structural Queries lässt sich ein Matching ausgehend von der Struktur des Inhalts durchführen. Dieser Query-Typ eignet sich sehr gut für hierarchische Schemata wie sie für HTML oder XML-Dokumente (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 149ff.]) existieren. Zu den am häufigsten anzutreffenden Strukturen (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 106ff.]) zählen:

- Fixed Structure
- Hypertext Structure
- Hierarchical Structure

Context Queries

- Phrase:
  Die Anfrage besteht aus einem Satz als Sequenz aufeinanderfolgender Tokens. Unter Umständen wird eine Normalisierung (z.B. Stop-Wort-Eliminierung) durchgeführt (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 101]). [KOWALSKI 1997, S. 29] bezeichnet diese Technik als *Contiguous Word Phrases* (CWP) und beschreibt damit eine genau festgelegte Zeichenkette, bei der keine Normalisierung durchgeführt wird.

- Proximity
  Proximity Search ist eine weniger restriktive Form als das Phrase Query (s.o.). Suchworte müssen hier keine direkten Nachbarn sein, sondern können auch durch die Angabe eines maximalen und minimalen Abstands zueinander identifiziert werden (cf. [KOWALSKI 1997, S. 28]).

Natural Language ([KOWALSKI 1997]: Natural Language Queries)

Diese Query Language erlaubt die Formulierung einer Anfrage in natürlicher Sprache. In vielen Fällen kann diese Formulierung durch eine morphologische und/oder Syntax-Analyse auch in eine Boolesche Anfrage transformiert werden. Die Schwierigkeit besteht in der Erkennung von Verneinungen (Negation) (cf.[KOWALSKI 1997, S. 34]).

Pattern Matching ([KOWALSKI 1997]: Term Masking)

Wie der Name bereits andeutet, wird ein Matching der Suchformulierung anhand spezifizierter Muster vollzogen. Die gebräuchlichsten Muster (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 105]) sind:

- Ganze Wörter
- Prefixes ([Kowalski 1997]: Prefix Search)
- Suffixes ([KOWALSKI 1997]: Suffix Search)
- Substrings ([KOWALSKI 1997]: Imbedded String Search)
- Ranges
- Reguläre Ausdrücke (*Regular Expressions*)
- Allowing Errors[2] ([KOWALSKI 1997]: Fuzzy Searches)

Soundex

Das Prinzip von Soundex basiert darauf, einen Ähnlichkeitsvergleich aufgrund einer phonetischen Analyse zwischen den Bestandteilen von Query und Dokument durchzuführen. Es wird unter diesem Namen auch in der Oracle Datenbank (cf. Kap. 4.3.3 und [LONEY; KOCH 2001, S. 1211]) eingesetzt.

3.1.3 Query Operations

Query-Operationen bauen auf den Query Languages auf; [BAEZA-YATES; RIBEIRO-NETO 1999, S. 117] unterstreicht deshalb ihre Verwendung zur Erweiterung und damit Aufwertung der ursprünglichen Query-Formulierung des Benutzers, da dieser in den seltensten Fällen in der Lage ist, eine ideale Formulierung seines Informationsbedürfnisses zu liefern. Aus diesem Grunde werden diese Operationen auch als *Query Reformulation Strategies* (cf. [BAEZA-YATES; RIBEIRO-NETO 1999,

[2] Suche nach Worten mit einer ähnlichen Buchstabenkombination. Der Einsatz erfolgt meist zur Kompensierung von Tippfehlern o.ä.; z.B. bei durch OCR (*Optical Character Recognition*) eingelesenen Daten.

S. 118]) bezeichnet. Auch [GROSSMAN; FRIEDER 1998] unterstreichen den komplementären Charakter dieser Operationen, indem er sie als 'Utilities' bezeichnet. Bei ihrer Verwendung zur Erweiterung des Query lassen sich drei Ansätze der Durchführung unterscheiden (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 117]):

1. In Interaktion mit dem Benutzer (User Relevance Feedback)
2. Automatisch und auf den Dokumenten der Retrievalmenge aufbauend (lokale Analyse)
3. Automatisch und auf der gesamten Kollektion aufbauend (globale Analyse)

Zu den eingesetzten Mitteln (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, Kap. 5] und [KOWALSKI 1997, S 32, 125ff.]), mit deren Hilfe sich 'Erweiterungen' und 'Neugewichtungen'[3] an einem Query vornehmen lassen, gehören:

- Clustering
- Thesaurus
- Context Analysis / Concept Catalogue[4]
- Normalization Features, i.e. Date und Number Recognition

Die Einbeziehung des Benutzers ist nicht nur beim User Relevance Feedback von Bedeutung, sondern spielt bei jeglicher Art der Erweiterung, auch der automatischen, eine Schlüsselrolle. Im Idealfall sollte der Benutzer jederzeit in Entscheidungen einbezogen werden können, wenn es z.B. darum geht, um wie viele Worte ein Query erweitert werden soll, d.h. bis zu welcher Tiefe (*Level*) verwandte Begriffe (z.B. Synonyme eines Thesaurus) in die Erweiterung einbezogen werden sollen. Wie in Kap. 2.4.1 bereits erwähnt wurde, kann das Ausmaß der Erweiterungen erheblichen Einfluss auf Precision- und Recall-Werte haben. Der Benutzer kann in diesen Fällen hilfreich bei der Bereinigung der Ergebnisse der automatischen Analysen sein, indem er Worte oder Zusammenhänge aussortiert, die nicht seinem Informationsbedürfnis entsprechen.

3.2 Post-Result-Operationen

Zu den Post-Result-Operationen zählen solche, die auf den Ergebnissen der Matching-Funktion, i.e. den Rankings aufbauen. Zu diesen Operationen gehören beispielswiese die folgenden.

[3] *Query Expansion* und *Term Reweighting* (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 118ff.].

[4] Hierarchische Kataloge (cf. [KOWALSKI 1997, S. 32f.]) werden u.a. bei vielen Web-Portalen wie Yahoo (http://www.yahoo.de, Zugriff: 18-August-2002) als Einstiegshilfen angeboten.

Relevanz-Normalisierung

In Kap. 2.5 wurde bereits auf die Verschmelzung einzelner Ergebnislisten zu einem Gesamtranking eingegangen. Hierfür ist es notwendig, alle Relevanzen durch Normalisierung auf einen gemeinsamen Werteraum abzubilden. Dieser Aufgabe kann eine spezielle Komponente nachkommen, die im Retrievalprozess nach dem Matching eingeordnet ist.

Re-Ranking

Bei der Fusion einzelner Rankings können zusätzliche Einflussfaktoren eine Neugewichtung der einzelnen Treffer bewirken, die sich auf die endgültige Position des Treffers im Gesamt-Ranking auswirkt. Die in Kap. 2.5 erwähnte Lern-Komponente kann z.B. solche Faktoren in Form von Gewichten zur Verfügung stellen, um die Treffer einzelner Rankings bei der Fusion auf- bzw. abzuwerten.

Highlighting

Eine Highlighting-Komponente durchsucht die gefundenen Dokumente nach den im Query spezifizierten Wörtern und markiert diese. Auf diese Weise kann der Benutzer die Suchworte im Text auf Anhieb lokalisieren und somit schneller einen Eindruck von der Relevanz des Dokuments gewinnen (cf. [KOWALSKI 1997, S. 37]).

4 Aufgaben der IR-Systeme

In diesem Kapitel steht die praktische Umsetzung von IR-Verfahren im Zusammenhang mit Aufgaben der Speicherung und Repräsentation von Daten im Vordergrund. Je nach IR-System ist auch die Unterstützung dieser Aufgaben und Verfahren unterschiedlich stark ausgeprägt.

> "An Information Retrieval System is a system that is capable of storage, retrieval, and maintenance of information." [KOWALSKI 1997, S. 2]

> "[...] the primary goal of an IR system is to retrieve all the documents which are relevant to a user query while retrieving as few non-relevant documents as possible." [BAEZA-YATES; RIBEIRO-NETO 1999, S. 2]

Zu den bekanntesten IR-Systemen gehören INQUERY[1], das kommerzielle RetrievalWare[2] oder auch das frei erhältliche SMART[3]. Sie stellen hochentwickelte Systeme dar, wobei die meisten – wie im Fall von SMART – auf das Retrieval von Textinformationen optimiert sind oder wie RetrievalWare auch multimediale Informationsobjekte unterstützen und einen integrierten Ansatz wählen. Sie basieren auf einem der in Kap. 2.2 dargestellten IR-Modelle oder einer Variante und implementieren meist zusätzliche Mechanismen wie die Unterstützung für natürlichsprachige Anfragen, Stemming oder Relevance Feedback (cf. Kap. 3), die eine Verbesserung der Retrievalqualität zum Ziel haben.

[1] INQUERY ist ein am Center for Intelligent Information Retrieval (CIIR) der University of Massachusetts at Amherst entwickeltes IR-System. Es basiert auf zwei Bayes'schen Netzwerken, wobei eines zur Repräsentation der Dokumente und das andere zur Repräsentation der Anfragen verwendet wird. Bayes'sche Netzwerke zählen zu den probabilistischen Modellen (cf. 2.2). http://ciir.cs.umass.edu/demonstrations/InQueryRetrievalEngine.html (Zugriff: 10-Juni-2002)

[2] http://www.convera.com/Products/products_rw.asp (Zugriff: 12-Juni-2002)

[3] Eines der ersten und für lange Zeit erfolgreichsten IR-Systeme ist das von Gerald Salton entwickelte SMART, das auf einem Vektor-Modell basiert. Die Quellen sind frei erhältlich unter der URL ftp://cs.cornell.edu/pub/smart (Zugriff: 10-Jun-2002)

> "The general objective of an Information Retrieval System is to minimize the overhead of a user locating needed information. Overhead can be expressed as the time a user spends in all of the steps leading to reading an item containing the needed information [...]." [KOWALSKI 1997, S. 4]

Im Falle paralleler, verteilter Architekturen wie bei Meta-Suchmaschinen, in der Anfragen an multiple IR-Systeme weitergeleitet werden, lässt sich das Kriterium *Overhead* allerdings nur bedingt anwenden. Man kann sich vorstellen, dass beim Einsatz multipler IR-Systeme die ersten Ergebnisse bereits vorliegen, während man auf andere noch wartet. Bekommt der Informationssuchende nun diese Teilergebnisse direkt zur Einsicht oder muss er so lange warten, bis alle Teilergebnisse vorliegen? In solchen Fällen kann es zur Aufgabe des Benutzers gehören, eine Angabe bezüglich der maximalen Wartezeit zu machen, nach deren Überschreitung die zu diesem Zeitpunkt vorliegenden Resultate zurückgeliefert werden sollen. Später eintreffende Ergebnisse werden verworfen oder zur späteren Abholung zurückgelegt. Diese Angabe der maximalen Wartezeit erfüllt auf zeitlicher Ebene dieselbe Funktion, i.e. Abbruchbedingung wie die Angabe eines *Cut-Off*-Wertes zur quantitativen Eingrenzung der Ergebnismenge[4].

Die Grundlage für ein IR-Modell bildet, wie in Kap. 2.2 angesprochen, ein Framework, das u.a. Datenstrukturen zur Speicherung der Informationsobjekte (Dokumente) und ihrer logischen Repräsentationen umfasst. [KOWALSKI 1997, Kap. 4] unterteilt in diesem Zusammenhang in zwei Strukturen, von denen die eine in ihrer Funktion mit der eines Dokumentenmanagers verglichen wird. Sie enthält die Dokumente in ihrem originalen Zustand und regelt Speicherung und Zugriff während die andere Struktur der logischen Sicht auf die Dokumente[5] entspricht und mit deren Hilfe das Matching durchgeführt wird. Die Elemente der logischen Sicht (z.B. Terme) können von unterschiedlicher Komplexität sein und z.B. zusätzlich die Position des Terms innerhalb des Dokuments enthalten, um Structured Queries (cf. Kap. 3.1.2) zu ermöglichen.

[4] Die Verwendung einer maximalen Wartezeit ist z.B. bei der Meta-Suchmaschine MetaGer möglich; http://www.metager.de (Zugriff: 20-Juli-2002)

[5] Am häufigsten werden für die logische Sicht invertierte Strukturen wie *Inverted Files* verwendet (cf. [SALTON; MCGILL 1983, S. 19f.], [KOWALSKI 1997, S. 76ff.] und [BAEZA-YATES; RIBEIRO-NETO 1999, S. 224f.]).

4.1 Heterogenität der Datenstrukturen

In Kap. 2 wurde bereits auf Strukturiertheit in Bezug auf den Inhalt von Dokumenten eingegangen. In diesem Abschnitt zielt Strukturiertheit auf die Speicherorganisation der Dokumente selber ab. Hierbei lassen sich formatierte und unformatierte Modelle unterscheiden, die bezüglich des Datenkonzeptes, der Datenbereitstellung und der Datenverwaltung unterschiedliche Hilfsmittel verwenden.

Strukturierte Datenbestände werden dadurch charakterisiert, dass für ihre abgelegten Daten ein gemeinsames Schema verwendet wird. Meistens handelt es sich um Datenbanksysteme (relational oder objektorientiert), die intern Suchmechanismen wie Volltextsuche, Indexverwaltung etc. implementieren. Die Manipulation und Verwaltung der Datenbestände und der Zugriff auf Suchfunktionen erfolgt in diesen Fällen über das jeweilige DBMS.

Unstrukturierte Datenbestände können dagegen als 'lose' Datensammlungen verstanden werden. Dazu gehören vor allem das WWW mit seinen Milliarden Web-Seiten oder lokale Dateisammlungen (Text, Audio, Video), die lediglich per Browsing-Technik durchstöbert werden können. Zusammenhänge zwischen den Dokumenten lassen sich erst durch Betrachtung oder – im Fall von Hypertext – durch Verfolgung von Verweisen erschließen. Um diese Datenbestände für das Retrieval nutzbar zu machen, i.e. um eine logische Sicht zu erstellen, bedarf es zusätzlicher Indexierungsprozesse wie es durch Suchmaschinen im Internet heute praktiziert wird. Hervorzuheben ist in diesem Zusammenhang die Indexierungsweise der Suchmaschinen Google[6] und Teoma[7]. Ihre Indexierungstechnik verwendet als ein Kriterium für das Ranking einer Web-Seite die Linkstruktur[8]. Für die Relevanz eines Dokuments bedeutet dies, je mehr Links anderer Web-Seiten auf dieses Dokument verweisen, desto bedeutender wird das Dokument eingeschätzt. Darüberhinaus bemüht sich Google herauszustellen, dass durch das Page-Rank-Verfahren eine Termindexierung und -gewichtung[9] auch auf den Link-Seiten durchgeführt wird, was die Retrievalqualität zusätzlich erhöhen soll. Die Menge der in Web-Suchmaschinen indexierten Dokumente überschneidet sich allerdings kaum. Ihre Indizes korrespondieren mit weniger als 2% (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 69]). Dies ist ein Grund, warum Meta-Suchmaschinen besonders in Bezug auf das World Wide Web (WWW) als vielversprechendes Suchinstrument gelten. Diese Tatsache verdeutlicht in diesem Zusammenhang noch einmal den möglichen Nutzen durch das bereits vorgestellte Konzept der Mehrfachdeskribierung (cf. Kap. 2.5), bzw. der

[6] www.google.de (Zugriff: 23-Juli-2002)
[7] www.teoma.com (Zugriff: 23-Juli-2002)
[8] Eine Einführtung in das Page-Ranking aufgrund der Linkstruktur bietet die URL http://www.google.com/technology/index.html (Zugriff: 15-Jun-2002).
[9] Zu *Term Frequency* (TF) und *Inverse Document Frequency* (IDF), cf. [KOWALSKI 1997, S. 103ff.]

Metasuche.

In Bezug auf das MIMOR-Modell (cf. Kap. 2.5) ist es im Sinne der Mehrfachdeskribierung, eine größtmögliche Vielfalt unterschiedlicher Datenbestände zu integrieren. Zur Einordnung wird folgende Unterteilung vorgenommen. Die Angabe in Klammern spezifiziert mögliche Suchhilfsmittel.

1. Unformatierte Datenbestände
    - WWW ([Meta-] Suchmaschinen, Kataloge, Agenten)
    - Dokumentsammlungen (lokale Suchmaschinen, Kataloge)
2. Formatierte Datenbestände
    - DBMS (Datenbankschnittstelle)
    - IR-Systeme (eigene Such-Mechanismen)

Im folgenden erfolgt eine Konzentration auf formatierte Datenbestände. Hierzu sollen Unterschiede und Gemeinsamkeiten von IR-Systemen und DBMS genauer betrachtet werden.

4.2 IR-System vs. DBMS

Die in der Vergangenheit gezogene Trennlinie zwischen IR-Systemen und Datenbankmanagementsystemen (DBMS) hat aus heutiger Sicht ihre Berechtigung verloren.

DBMS werden traditionell für das Retrieval von Faktenwissen (Data Retrieval) eingesetzt, da sich ihre Datenstruktur[10] ausschließlich für das Exact-Match-Paradigma (cf. Kap. 2) eignet. Auch die für DBMS verwendeten Anfragesprachen wie SQL lassen i.d.R nur ein Retrieval nach diesem Paradigma zu. Ihre Inhalte unterliegen strengen Restriktionen, die in Form von Datentypen und Integritätsregeln ausgedrückt sind.

Während die Ausrichtung auf das Data Retrieval bei traditionellen DBMS zutrifft, finden sich heutzutage bei vielen auf dem Mark erhältlichen Datenbankprodukten zusätzliche Features, die die Verwaltung und Erfassung unstrukturierter Dokumente oder multimedialer Objekte erlauben (cf. Kap. 4.3.2). Eine weitere Entwicklung hat in Bezug auf die Match-Paradigmen bei DBMS stattgefunden. Proprietäre Erweiterungen (cf. Kap. 4.3.1) der deklarativen Anfragesprache SQL erlauben beispielsweise in vielen

[10] DBMS als reine Data-Retrieval-Lösungen basieren auf einem Datenmodell bestehend aus einer *Data Definition Language* (DDL) und einer *Data Manipulation Language* (DML) (cf. [KEMPER; EICKLER 1999, S. 19]).

relationalen DBMS die Einbeziehung von Unsicherheit und Vagheit über schon lange in der DBMS-Welt verwendete Mechanismen wie Pattern-Matching (cf. Kap. 3.1.2) hinaus.

Diese Mechanismen und Erweiterungen sollen im folgenden genauer betrachtet werden. Dazu wird auch Bezug auf aktuell erhältliche DBMS genommen.

4.3 DBMS als Basis eines IR-Systems

In diesem Kapitel soll aufgezeigt werden, wie mit Hilfe handelsüblicher DBMS und ihrer Erweiterungen die Integration von Text- und Fakten-Retrieval vollzogen werden kann.

> "[...] there are several proposals that extend SQL [...] to allow full-text retrieval. Among them we can mention proposals by leading relational database vendors such as Oracle and Sybase [...]" [BAEZA-YATES; RIBEIRO-NETO 1999, S. 108]

4.3.1 Erweiterungen des SQL-Standards

SQL ist eine deklarative Programmiersprache, die sich sehr gut für Abfragen in Form von Ad-hoc-Queries eignet. In der Praxis setzen auf DBMS meist sehr komplexe Anwendungsprogramme auf, deren Programmierung den Einsatz von Schleifen, Rekursion oder Prozeduren verlangt. Da eine deklarative Programmiersprache wie SQL für diese aber keine Unterstützung bietet, müssen die Konstrukte der deklarativen Sprachen in die klassische (prozedurale) Programmiersprache der Anwendung eingebettet werden.

Eine andere Lösung besteht darin, eine erweiterte deklarative Programmiersprache zu verwenden, die in vielen Datenbanksystemen als proprietärer Bestandteil existiert. Der Nachteil liegt hier eindeutig in der Bindung an eine spezielle Lösung, durch die man sich in eine Abhängigkeit von Sprache, bzw. Hersteller begibt. Ein Beispiel für eine solche Sprache ist das von Oracle entwickelte und verwendete PL/SQL (cf. Kap. 4.3.3).

Viele DBMS haben in den letzten Jahren zusätzlich einen Mechanismus adoptiert, der die Ausführung von Programmcode unterschiedlicher Programmiersprachen innerhalb der Datenbank ermöglicht. Es handelt sich um in der Datenbank gespeicherte Funktionen einer prozeduralen Sprache (PL[11]), die vom DBMS ausführbar sind, indem sie sich wie Methoden aufrufen lassen. Während Oracle neben dem prozeduralen PL/SQL auch die

[11] PL = Procedural Language

Ausführung von Java-Klassen über gespeicherte Funktionen erlaubt, ist die Unterstützung bei anderen DBMS wie PostgreSQL (cf. Kap. 4.3.4) auf prozedurale Sprachen wie C, Tcl oder Perl begrenzt.

Für RECOIN kann geschlussfolgert werden, dass DBMS mit Hilfe gespeicherter Prozeduren einen erweiterten Funktionsumfang zur Verfügung stellen können, der über das reine Data Retrieval weit hinausgeht. Auf der einen Seite kann das Retrieval weiterhin über (standardisierte) deklarative Sprachkonstrukte erfolgen. Auf der anderen Seite lässt sich über gespeicherte Prozeduren der volle Funktionsumfang einer prozeduralen Programmiersprache ausnutzen. Dadurch lassen sich alle Besonderheiten eines speziellen Datenbanksystems ausschöpfen. Die Entwicklung gespeicherter Prozeduren, die Parameter des Text- und des Fakten-Retrievals bei Anfrage und Ranking berücksichtigen, wird dadurch möglich.

4.3.2 Erweiterte Datentypen

Fast alle gängigen DBMS erlauben heutzutage die Deklaration von binären Datentypen sehr großer Länge. Diese Datentypen werden im Datenbankbereich einfach als *Large Objects* (LOBs) bezeichnet (cf. [LONEY; KOCH 2001, S. 603ff.]) und in weitere Typen unterschieden. *Binary Large Objects* (BLOBs) beispielsweise erlauben die Speicherung binärer Daten wie sie bei Graphiken und Audio- und Videodokumenten anfallen.

Bei objektorientierten Datenbanken lassen sich Datentypen selbst definieren. Es "wurde von vornherein auf eine enge Anbindung an Programmiersprachen geachtet. Das Typsystem der Programmiersprache und das Datenmodell des Datenbanksystems sind eng integriert. Das Datenbankschema muss nicht mit der Schemadefinitionssprache (ODL = Object Definition Language) beschrieben werden, sondern kann mit einem erweiterten Typsystem der jeweiligen Programmiersprache definiert werden." [BALZERT 1996, S. 690]

Damit bei traditionellen DBMS sowohl unstrukturierte Textdokumente als auch binäre Daten mit in die Indexierung und damit das Matching einbezogen werden können, ist es notwendig, zusätzliche Komponenten – z.B. zur Volltext-Indexierung – einzusetzen, die ein traditionelles DBMS um zusätzliche Funktionen erweitern. Diese Erweiterungen werden von jedem DBMS in unterschiedlichem Ausmaß umgesetzt. Einige Beispiele werden in die folgenden Abschnitten vorgestellt.

4.3.3 Die Oracle Datenbank

Die grundsätzlichen Erweiterungen in Form der Einbettung von Programmiersprachen und Datentypen wurden bereits vorgestellt. Mit PL/SQL existiert in Oracle einerseits eine eigene prozedurale Sprache, die gegenüber dem deklarativen SQL-Standard das Schreiben kleiner Programme inkl. Schleifen und Verzweigungen erlaubt. Andererseits erweitert Oracle den SQL-Sprachumfang um proprietäre Konstrukte, die nicht Teil des SQL-Standards sind. Ein Beispiel hierfür ist der *CONNECT BY*-Befehl, mit dem sich ohne größeren Aufwand der Umfang einer rekursiven Relation, die sog. transitive Hülle, berechnen lässt (cf. [KEMPER; EICKLER 1999, S. 116]).

Oracle verwendet weiterhin sehr mächtige Komponenten, die in einer Software-Suite mit dem Namen *Intermedia* zusammengefasst sind. Beispiele für Komponenten dieser Suite sind Intermedia-Text (cf. [LONEY; KOCH 2001, Kap. 24]) für das Text-Retrieval oder Intermedia-Audio und -Image. Die Verwendung von Intermedia-Text erlaubt den Aufbau von Volltext-Indizes unter Verwendung von Stemming, Stoppwortlisten und weiterer Parameter. Darauf aufbauend wurden SQL-Erweiterungen eingeführt, mit denen sich Rankings erzeugen lassen. Hierzu existiert der *CONTAINS*-Befehl der in Kombination mit verschiedenen Operatoren unterschiedliche Query Languages (cf. Kap. 3.1.2) umsetzen kann. Zu diesen Operatoren gehört z.B. *NEAR*, mit dem sich eine Proximity Search durchführen lässt. Weitere Möglichkeiten sind Fuzzy Matches, Soundex und Wortstammerweiterung. Dabei lassen sich diese Operatoren kombinieren und ihre Reihenfolge der Auswertung lässt sich durch Setzen von Klammern beeinflussen.

In der Architektur von Oracle ist eine eindeutige Dominanz des DBMS-Prinzips zu erkennen, bei dem die Unterstützung für unstrukturierte Daten und Vagheit lediglich durch Erweiterungen umgesetzt wird.

4.3.4 MySQL und PostgreSQL

MySQL[12] und PostgreSQL[13] gehören zu den frei erhältlichen DBMS, die unter einer Open-Source Lizenz weiterentwickelt werden. Obwohl ihr Funktionsumfang bei weitem nicht an den kommerzieller Datenbanken wie Oracle oder DB2 von IBM heranreicht, so sind in den neueren Versionen zahlreiche Fähigkeiten hinzugekommen, die sie für den Einsatz in RECOIN interessant machen. Dazu gehört die bereits erwähnte Unterstützung binärer Datentypen, gespeicherter Prozeduren und der Aufbau von Volltext-Indizes.

[12] http://www.mysql.org (Zugriff: 13-Juni-2002)
[13] http://www.postgres.org (Zugriff: 28-Juni-2002)

4.4 Neue Ansätze der Datenspeicherung

Dieser Abschnitt stellt unterschiedliche Prinzipien der Datenspeicherung in DBMS vor. Es wird vorausgesetzt, dass die Prinzipien der relationalen Datenspeicherung, die den De-facto-Standard bei DBMS darstellen, bekannt sind. Vorgestellt werden sollen neuere Ansätze, wie objektorientierte DBMS und native XML-Datenbanken.

4.4.1 Objektorientierte Datenbanksysteme

Im Gegensatz zu relationalen Datenbanksystemen[14] werden Objekte in OO-Datenbanken in unveränderter Form, also nicht transformiert in Tabellen, gespeichert. Analog liegt dem Schema kein relationales, sondern ein objektorientiertes Modell zugrunde. Die Sprache, mit der dieses Modell erstellt wird, bezeichnet man allgemein als *Object Definition Language* (ODL), die Abfragesprache als Objekt Query Language (OQL) (cf. [BALZERT 1996, S. 677ff]). Einen Ansatz zur Standardisierung dieser Sprachen stammt von der *Object Database Management Group* (ODMG[15]). Der ODMG-93-Standard ist im Januar 2000 als neue Version 3.0 eingeführt worden. Ein weiterer, neuerer Spezifikationsvorschlag der ODMG behandelt die Einbindung von Java in den ODMG-93-Standard und ist auf den Namen *Java Data Objects* (JDO) getauft worden.

> "Die Vorteile der Objektorientierung liegen in der Nähe zur und gleichzeitigen Unabhängigkeit von der Implementierungsebene." [VOSS; GUTENSCHWAGER 2001, S. 182]

Die Objektorientiertheit erfüllt viele der Ansprüche, die an heutige Software gestellt werden, zu denen beispielsweise die folgenden gehören:

- Kurze Entwicklungszeiten
- Wiederverwendbarkeit
- Abbildung komplexer Strukturen und Prozesse
- Modularer Aufbau (Baukastenprinzip)
- Flexibilität

[14] Für eine Gegenüberstellung der Eigenschaften relationaler versus objektorientierter Datenbanken cf. [BALZERT 1996, S. 681, Tab. 3.4-1].
[15] http://www.odmg.org (Zugriff: 22-Juli-2002)

4.4.2 Native XML Datenbanksysteme

Dadurch, dass die Dokumente und Topics der in Kap. 2.4.2 vorgestellten Referenzkollektionen SGML-formatiert vorliegen und die XML[16] eine Untermenge der SGML darstellt, soll an dieser Stelle auf die zunehmende Verbreitung dieser Markup-Sprache eingegangen werden.

Fast alle gängigen Datenbankhersteller, darunter Oracle, IBM und Microsoft, bieten für ihre Produkte Erweiterungen an, um XML-Daten auszutauschen und zu speichern. Allerdings handelt es sich dabei 'nur' um Ad-hoc-Lösungen, die XML-Funktionalitäten auf einer anderen, meist relationalen, Datenstruktur aufsetzen. Diese Formen von DBMS werden aus diesem Grund als XML-Enabled DBMS bezeichnet. Die Daten selber werden lediglich als BLOBs (cf. 4.3.2) oder CLOBs (*Character Large Objects*) in einer Tabelle abgelegt und indiziert oder per Volltextsuche abgefragt.

Im Gegensatz dazu ermöglichen es native XML Datenbanksysteme die Vorzüge der XML voll auszuschöpfen. Der Begriff Native XML DBMS (XDBMS) wird verwendet, um hervorzuheben, dass diese DBMS ein vollständig auf XML basierendes Datenmodell verwenden. Die angesprochenen Vorzüge bei der Verwendung von XML liegen z.B. in der Verfügbarkeit spezieller Anfragesprachen wie XPath (cf. [GOLDFARB 2002, S. 940]) und Xquery (cf. [GOLDFARB 2002, S. 967]) zur Lokalisierung von Dokumenten und Dokumentelementen.

XML-Dokumente weisen eine hierarchische Struktur auf, die von Anfragesprachen wie der SQL nicht effizient traversiert werden kann. Zwar existieren auf Seiten von DBMS wie Oracle und DB2 proprietäre Erweiterungen (cf. Kap. 4.3.1) der SQL , die dies ermöglichen, doch sind diese weder allgemein gültig noch standardisiert. Native XDBMS bieten hier mit der Verwendung von XPath einen entscheidenden Vorteil. Mit Hilfe von XPath-Ausdrücken lassen sich Hierarchien nicht nur in Richtung der Kind- oder Eltern-Elemente durchlaufen, sondern auch horizontal[17]. Weitere Neuerungen, die als sog. Working Drafts beim W3C[18] kurz vor ihrer offiziellen Standardisierung stehen, sind die auf XPath aufbauende XQuery oder XQL (*XML Query Language*). Ihre Fähigkeiten in Bezug auf das Fakten-Retrieval reichen inzwischen an die relationaler DBMS heran. Der große Vorteil ist, dass die Fakten nicht aus den Dokumenten 'herausgelöst' und in ein relationales Datenmodell eingefügt werden müssen.

[16] SGML (Standardized General Markup Language) und XML (Extensible Markup Language) werden in Kap. 5.4 noch einmal vorgestellt.

[17] Das bedeutet, dass sich mit einer simplen 'Pfadangabe' alle Elemente einer 'Ebene' erreichen lassen (cf. [GOLDFARB 2002, S. 955]), z.B. alle Autoren von Dokumenten, deren Name mit 'K' beginnt. Mit der SQL ist die Extraktion dieser Informationen mit größerem Aufwand verbunden.

[18] World Wide Web Consortium; http://www.w3c.org (Zugriff: 25-Aug-2002)

5 Metadaten

Zum Verständnis, wie Metadaten in RECOIN eingesetzt werden und welche Funktion sie ausüben können, erscheint es sinnvoll, sich vorher eingehender mit ihnen zu befassen.

5.1 Was sind Metadaten?

Metadaten dienen der Beschreibung von Objekten und können auch selbst wieder Objekte repräsentieren. Im Bibliothekswesen werden Metadaten zur Identifizierung und Lokalisierung von Informationsobjekten eingesetzt. Sie existieren dort in Form von Kategorisierungen, Standortinformationen und Angaben zu Dokumentmerkmalen wie Autor oder Publikationsdatum.

> "[...] metadata is data about the data'."
> [BAEZA-YATES; RIBEIRO-NETO 1999, S. 142]

Der aus dem Griechischen stammende Begriff *meta* bedeutet 'über' oder 'hinter'. Traditionell werden Metadaten deshalb als *Daten über Daten* verstanden. Metadaten steuern die semantische Komponente bei, mit der der Transport und die Bearbeitung von Daten in vielen Fällen überhaupt erst möglich gemacht wird.

Weitere Beispiele für Metadaten sind die Datentypen eines relationalen Datenbankschemas, mit denen sich der Spalteninhalt einer Tabelle beschreiben lässt oder die Angaben im Kopf einer Email, die Auskunft darüber geben, welche Daten der Absender, welche das Empfangsdatum und welche der Betreff sind.

5.2 Anforderungen an Metadaten

Beim Umgang mit Metadaten lassen sich besonders im Zusammenhang mit elektronischen Informationsobjekten und der elektronischen Speicherung besondere Ansprüche stellen.

- Metadaten sollten maschinenlesbar sein, damit sie genau wie die Daten, die sie beschreiben, elektronisch verarbeitet und ausgewertet werden können.

  "One application's metadata is another application's data"[1]

- Der Bezug der Metadaten zu den durch sie beschriebenen Objekten muss eindeutig sein. Unter strukturellen Gesichtspunkten bieten sich hier zwei Alternativen an: Metadaten können im beschriebenen Objekt selbst (*objektintrinsisch*) oder extern (*objektextrinsisch*) erstellt und mit dem zu beschreibenden Objekt verknüpft werden[2].

- Die Beschreibung von Objekten sollte durch die Verwendung eines standardisierten Metadatenvokabulars eingeschränkt werden. Das einheitliche Vokabular sichert die Gültigkeit der Metadaten auch in Zukunft und unterdrückt den Wildwuchs durch Eigendefinitionen. Um dennoch ein gewisses Maß an Freiheit zu gewähren, kann eine Unterteilung in obligatorische und fakultative Elemente erfolgen. Der Umfang und die Architektur des Vokabulars sollte derart sein, dass auch eine Beschreibung inhomogener Objekte möglich ist.

Metadaten können prinzipiell überall dort nutzbringend eingesetzt werden, wo informationstragende Objekte zielorientiert um strukturierte Informationen erweitert werden sollen. In RECOIN sollen sie zur Beschreibung der Objekte eingesetzt werden, die am Retrievalprozess beteiligt sind. An dieser Stelle stellt sich nun die Frage nach einer geeigneten Speicherung der Metadaten.

[1] Quelle: http://www.w3.org/People/EM/talks/www7/tutorial/part2/sld004.htm, Slide 4 (Zugriff: 23-Juni-2002)

[2] Die Sprache XML (cf. 5.4) erlaubt die Mischung von sog. *internal* und e*xternal subsets* (cf. [GOLDFARB 2002, S. 793])

5.3 Data Dictionaries und Repositories

Der Begriff *Repository* oder *Data-Dictionary*[3] stammt u.a. aus dem Bereich der Programmierung, wo ein Data Dictionary eine Sammlung von Beschreibungen der Objekte eines Datenmodells enthält. Der Nutzen des Data Dictionary ergibt sich aus der Tatsache, dass diese Informationen system- bzw. unternehmensweit zur Verfügung gestellt werden können. Projekte, die diese Datenobjekte in eigenen Modellen verwenden müssen, erhalten so immer ein konsistentes Bild des aktuellen Datenmodells.

Aus dem Englischen übersetzt bedeutet Repository so viel wie Depot, Lager oder schlicht Ablage. In seiner Bedeutung für die Informatik wurde der Begriff hauptsächlich durch das Software Engineering geprägt, wo ein Repository als Bestandteil von Entwicklungswerkzeugen wie CASE-Tools (C*omputer Aided Software Engineering*) zum Einsatz kommt. Da die Ergebnisse einzelner Entwicklungsphasen von nachfolgenden Werkzeugen benötigt werden, müssen sie in einer zentralen Entwicklungsdatenbank, dem *CASE-Repository* gespeichert werden. Die Aufgabe des Repository ist es u.a., geeignete Zugriffsmethoden über eine Schnittstelle zur Verfügung zu stellen und die im Entwicklungsprozess entstehenden Metadaten zu verwalten.

> "Der Zugriff auf ein Repository erfolgt immer über ein Informationsmodell." [NAUER 1991, S. 26]

> "Die objektorientierte Art der Datenmodellierung und des Zugriffs erweist sich dabei als geeignete Basistechnologie zur Reduzierung der Komplexitätsgrade und Kardinalitäten auzubauender Informationsstrukturen." [LJUBOJEVIC 1991, S. 44]

Neben der Frage der adäquaten Modellierung ergibt sich weiter die Frage nach der geeigneten Übertragung von Metadaten. Die Informationslogistik (cf. Kap. 2.1) verwendet in diesem Zusammenhang den Begriff der *Stickiness* von Informationen, um den Grad des Aufwands zu beschreiben, der notwendig ist, um zur Kontrolle oder Entscheidung notwendige Informationen zu transferieren. Je größer dieser Aufwand ist, desto 'klebriger' (*stickier*) die Informationen.

> "Die Stickiness von (Vereinbarungs- und Kontroll-) Informationen kann durch neue Technologien in der Datenhaltung und vor allem im Datenaustausch zum Teil gesenkt werden, i.e. es wird ein vereinfachter Zugriff auf

[3] Für eine Darstellung der Unterschiede zwischen Repository und Data Dictionary cf. [MYRACH 1994, S. 4ff]

> entsprechende Informationen ermöglicht.“ [VOSS; GUTENSCHWAGER 2001, S. 48]

Eine Möglichkeit zur Speicherung und Übertragung von Daten stellt die ***eXtensible Markup Language*** (XML) dar. Sie ist eine noch relativ junge Sprache, mit der sich die Datenhaltung und der Datenaustausch erheblich vereinfachen lassen.

5.4 XML als syntaktische Basis

XML ist keine reine Markup-Sprache wie HTML, sondern eine Meta-Sprache, deren Bedeutung im Strukturieren und Modellieren von Daten liegt, nicht in Präsentation und Layout (cf. [BAEZA-YATES; RIBEIRO-NETO 1999, S. 154f.]).

Zu den grundlegenden Merkmalen und Vorzügen der XML gehören die folgenden. Sie sind gleichzeitig ausschlaggebend dafür, dass die XML als bevorzugtes Speicher- und Transportmittel für (Meta-)Daten in RECOIN angesehen wird.

- Markup-Elemente dienen der einheitlichen Strukturierung und Repräsentation von Dokumenten (cf. [GOLDFARB 2002, S. 33, 38]).
- Die Kodierung (*Character Set*) der Sprache ist Unicode (UTF-8). Damit sind XML-Dokumente praktisch mit jedem Text-Editor bearbeitbar (cf. [GOLDFARB 2002, S. 36]).
- Der Sprachstandard ist erweiterbar (*extensible*) und erlaubt so die Definition eigener, domänenspezifischer Vokabularien (cf. [GOLDFARB 2002, S. 772ff.]).
- Mit Hilfe von Dokumenttyp-Definitionen (DTD) bzw. Schemata lässt sich die Struktur für eine Klasse von Dokumenten vorgeben. Die Einhaltung des Schemas (*Validity*) kann damit für jedes Dokument der Klasse überprüft werden (cf. [GOLDFARB 2002, S. 38]).

Wie HTML ist auch XML eine Untermenge der SGML (*Standard Generalized Markup Language*), die ihren Ursprung in den 70er Jahren hat und noch heute den De-facto-Standard für die Formatierung und den Austausch komplexer Dokumente größeren Umfangs darstellt.

Der Vorteil der Nutzung von XML zur Speicherung und Datenübertragung liegt u.a. darin, dass XML-Nachrichten keine kryptischen Nachrichten sind, sondern auch von Menschen gelesen und verstanden werden können. Sie sind damit auch in jedem Standard-Browser darstellbar.

6 Implikationen für RECOIN

Dieses Kapitel stellt in dieser Arbeit ein Zwischenfazit dar. Auf der Basis der vergangenen Kapitel sollen Konsequenzen für den Funktionsumfang von RECOIN gezogen und in neue Anforderungen umgewandelt werden.

6.1 Integration von Metadaten

Es stellt sich nun die Frage, inwiefern Metadaten als Bestandteil von RECOIN verwendet werden, bzw. welcher Nutzen aus ihnen gezogen werden kann. Aufgrund der Heterogenität der am Retrievalprozess beteiligten Objekte, bietet es sich an, diese mit Hilfe von Metadaten zu beschreiben. Als Beispiele für diese Objekte sollen Dokumente, Queries, IR-Systeme, etc. genannt werden.

Auf der einen Seite sollten die Metadaten genutzt werden können, um Objekte in RECOIN auf einer technischen Ebene zu beschreiben. Diese Metadaten sollen der Integration der Objekte auf Programmebene dienen. Hierzu gehören beispielsweise Informationen zu Treibern, Protokollen oder zur Benutzer-Authentifizierung, um nur einige zu nennen.

Auf der anderen Seite sollen mit den Metadaten die Charakteristika und Funktionen der Objekte in Bezug auf ihre Rolle im Retrievalprozess beschrieben werden. Ziel soll es sein, diese Metadaten der Lern-Komponente des MIMOR-Modells (cf. Kap. 2.5) zur Unterstützung der Relationierung zur Verfügung zu stellen. Für diese Metadaten können hier Länge und Typ des Informationsobjekts, IR-Modell, Indexierungsart und Anfragesprache eines IR-Systems als Beispiele genannt werden. Für die Evaluierung wichtige Charakteristika einzelner Objekte sollen sich auf diese Weise durch Metadaten ausdrücken, klassifizieren und in einem zentralen Schema verwalten lassen.

Sowohl für den internen als auch den externen Zugriff auf die Metadaten sollte in RECOIN eine Repository-Schnittstelle geschaffen werden.

6.2 Schnittstellenmanagement

Aufgrund der Tatsache, dass die technologische Weiterentwicklung sowohl im Hard- als auch im Softwarebereich mit sehr großem Tempo voranschreitet, ist es beinahe unmöglich, in Bezug auf die IT-Infrastruktur immer auf dem neuesten Stand zu sein. Verwendete Systeme durchlaufen immer einen Systemlebenszyklus, der dadurch bestimmt wird, wie sehr sich die umgebende Infrastruktur und die Anforderungen ändern oder welche Vorteile die Umstellung auf eine neue Technologie versprechen. Der Systemlebenszyklus ist – zwischen Anschaffung und Stilllegung – gekennzeichnet durch einen sich wiederholenden Wechsel von Adaption, Wartung und Pflege.

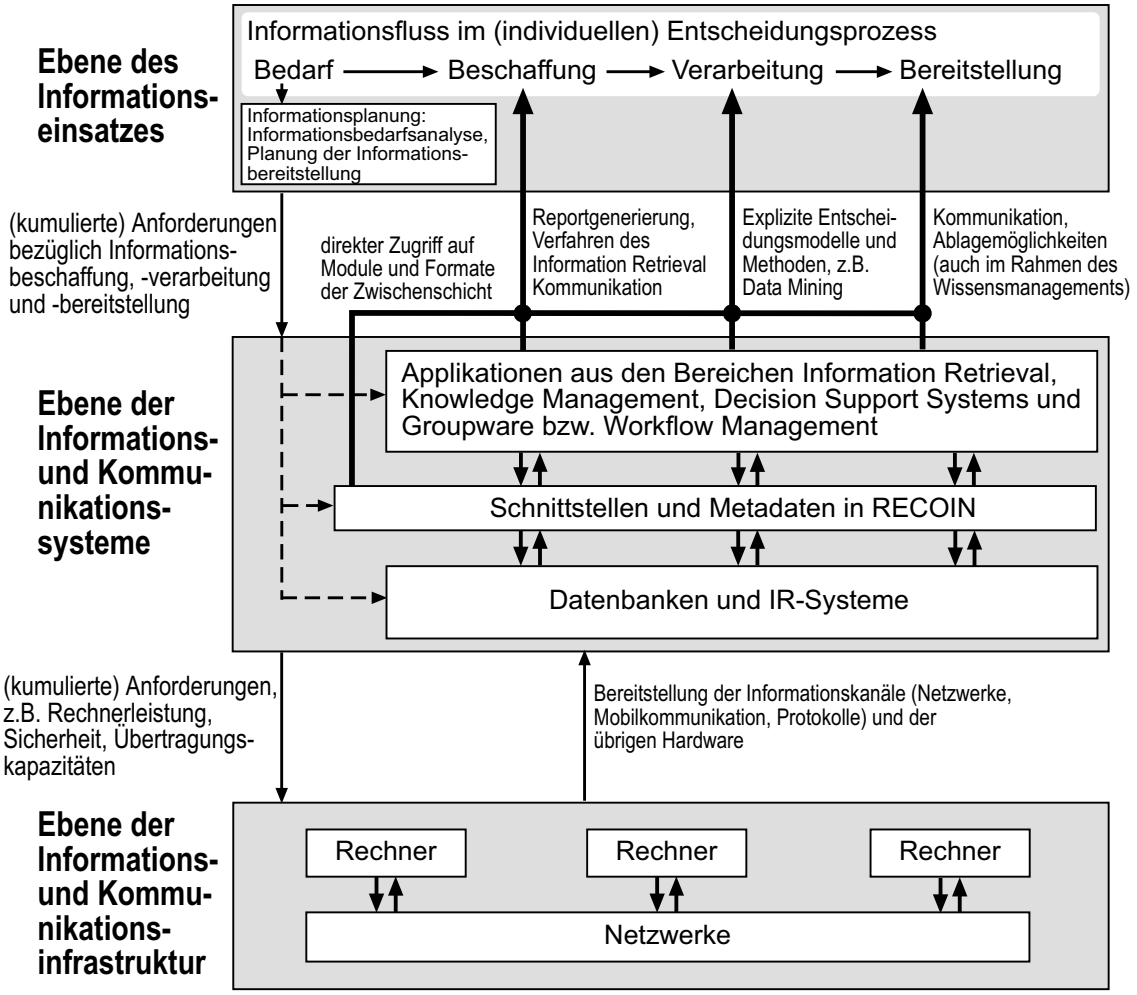

Abb. 6.1: Einordnung von RECOIN in das erweiterte Ebenen-Modell nach [VOSS; GUTENSCHWAGER 2001, S. 74]

> "Durch die asynchrone Entwicklung aller im Unternehmen befindlichen Systeme entsteht eine Heterogenität der eingesetzten Infrastruktur, insbesondere der Rechner und Anwendungen, die sich als ein wesentliches Problem darstellt." [VOSS; GUTENSCHWAGER 2001, S. 208].

Um die Kommunikation zwischen Systemen über den Wechsel einer Teilkomponente hinaus zu gewährleisten, werden gemeinsame Standards[1] für Schnittstellen und Protokolle definiert. Die Schnittstellen können als eine eigene Ebene zwischen miteinander kommunizierenden Systemkomponenten angesehen werden. Eine Einordnung dieser Schnittstellen-Ebene in ein erweitertes Ebenen-Modell ist in Abb. 6.1 dargestellt. Das Schnittstellenmanagement bietet ein hohes Potential zur Reduzierung von Transaktionskosten (cf. [VOSS; GUTENSCHWAGER 2001, S. 15, 46ff.]) an den Grenzen zwischen unterschiedlichen Anwendungssystemen, Abteilungen oder Unternehmen.

> "Für die Übertragung von Informationen kann eine Veränderung oder Zwischenlagerung des Informationsträgers erforderlich sein. [...]. Das sich ergebende Schnittstellenmanagement ist somit Teil der Informationslogistik [...]." [VOSS; GUTENSCHWAGER 2001, S. 314].

Dem Ebenen-Modell folgend verläuft jeglicher Informationsaustausch zwischen Software-Komponenten durch die Schnittstellenebene. Damit wäre es von Vorteil, auf diesem Teilstück der Übertragung den Informationsträger zu modifizieren, i.e. eine einheitliche 'Verpackung' zu verwenden. Diese Kapselung der Daten in ein internes Format bietet in diesem Anwendungsfall die Möglichkeit der Kommunikation über Protokollgrenzen hinweg. Abbildung 6.2 verdeutlicht dies am Beispiel von Benutzer-Applikationen, über die Anfragen gestellt werden, und IR-Komponenten, die sie beantworten. In seiner Rolle als Vermittler und Verteiler zwischen IR-Komponenten und informationsverarbeitenden Systemen muss RECOIN auch die Verantwortung für das Schnittstellenmanagement übernehmen. Dies ist eine der Voraussetzungen dafür, dass Software-Komponenten, die einen Bedarf an IR-Verfahren haben oder diese bereitstellen können, an RECOIN angeschlossen werden können, egal welches Protokoll oder welche Anfragesprache sie verwenden.

Weiterhin sollte RECOIN intern ein eigenes Transportmittel für die Daten verwenden, die durch die Schnittstellenebene fließen. Nur so kann RECOIN zum Bindeglied bzw. 'Dolmetscher' zwischen heterogenen Applikationen werden, die eigene Protokolle und Datentypen verwenden.

[1] Der Einsatz von Standards bietet u.a. die Vorteile der Investitionssicherheit und des verfügbaren Know-Hows durch eine große Anzahl von Entwicklern.

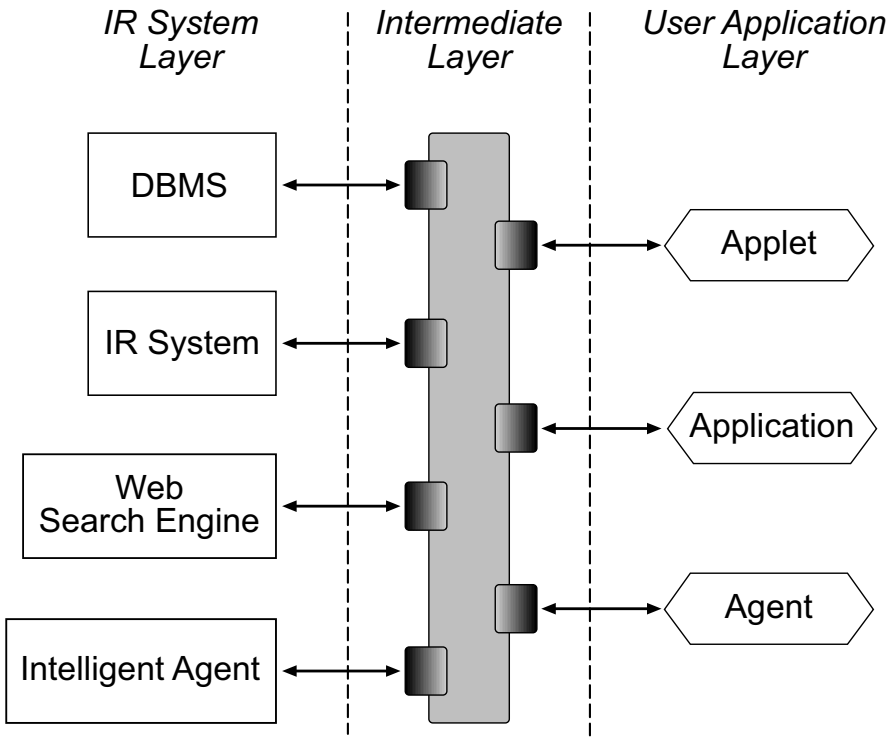

Abb. 6.2: Zwischenschicht als Protokollbrücke zwischen IR-Systemen und Anwendungssoftware

6.3 Modularisierung des Retrievalprozesses

Bei Betrachtung des Retrievalprozesses und der IR-Systeme in vorangegangenen Kapiteln ist deutlich geworden, dass neben dem Matching eine Vielzahl weiterer Sub-Prozesse bzw. Komponenten, beteiligt sind, die auf die Qualität des Retrievals Einfluss ausüben können.

> "Deshalb spielen m.E. Systeme oder Systemkomponenten eine beachtliche Rolle, welche die vorhandenen Gegebenheiten berücksichtigen und trotzdem mit einer Verbesserung der Qualität verbunden sind. Beispielhaft für diese Situation ist, Boolesches Retrieval mit einer intelligenten lokalen Komponente auszustatten, die zum einen die Suchformulierung unterstützt, zum anderen für die Sortierung des Outputs zuständig ist. [...] Dabei werden die Bereiche, die mit Unsicherheit behaftet sind (Überführung des Informationsbedürfnisses in eine Systemanfrage, Anordnung des Ergebnisses nach Relevanz, etc.), in eigenen Systemkomponenten erarbeitet, während meist die Datenhaltung und der Matchprozeß im exakten Booleschen System belassen wird." [WOMSER-HACKER 1996, S. 47].

Durch die Verwendung eines einheitlichen Formats den Datentransport innerhalb von RECOIN wird es nun möglich, autonome Komponenten und ihre Funktionen in den Retrie-

valprozess zu integrieren und sie somit anderen Systemkomponenten zur Verfügung zu stellen. Die Voraussetzung ist die Unterstützung des internen Datenformats durch diese Komponenten. RECOIN muss also die Verantwortung für die Einbindung der Komponenten auf Software-Ebene sowie für deren Zugriff auf den internen Datenstrom übernehmen. Weiterhin sollte die Verwaltung der Komponenten-Funktionen durch RECOIN realisiert werden, so dass eine Steuerung des Retrievalprozesses möglich wird. Das bedeutet, dass die angeschlossenen Komponenten für den Aufbau eines individuellen Retrievalprozesses ausgewählt und kombiniert werden können.

Um dabei möglichst flexibel zu bleiben und die Erweiterbarkeit von RECOIN zu sichern, ist ein strikt modularer Aufbau der Applikation notwendig.

6.4 Session-basiertes Retrieval

Ein besonders interessanter Aspekt bei IR-Systemen ist die Unterstützung von iterativen Anfragen besonders in Zusammenhang mit User Feedback (cf. Kap. 3.1.3). Aber auch wenn diese Unterstützung nicht von IR Systemen selber geleistet wird, so ist dennoch die Möglichkeit in Betracht zu ziehen, diese Funktionalität durch RECOIN zur Verfügung zu stellen.

Konkret bedeutet dies, dass Mechanismen für die Zuordnung von Benutzern oder Applikationen zu einer *Retrieval Session* zur Verfügung gestellt werden müssen. Außerdem sollte es möglich sein, mehreren Individuen gleichzeitig die Verwendung von RECOIN zu ermöglichen, was die Verwendung multipler Sessions bedeutet.

Bei iterativen Anfragen ist eine Session in mehrere Durchgänge (Cycles) unterteilt, wobei ein Retrieval Cycle der einmaligen Ausführung des Retrievalprozesses entspricht. Diese Tatsache verlangt, dass eine Session nicht automatisch mit der Beendigung des Retieval Cycle endet, sondern aktiv vom Benutzer oder einer Applikation beendet wird.

Dadurch, dass multiple Benutzer über multiple Sessions auf die an RECOIN angeschlossenen Komponenten zugreifen und ihre Funktionen nutzen, entsteht ein hohes Maß an Parallelität, dem die Gesamtarchitektur Rechnung tragen sollte.

6.5 Lastenheft

Aus den in diesem Kapitel hervorgegangenen Anforderungen an RECOIN wird nun ein Lastenheft nach dem Vorbild aus [BALZERT 1996, S. 58, Abb. 1.2-1] entwickelt, das in den folgenden Kapiteln die Grundlage für den Softwareentwicklungsprozess bildet.

Ziel eines Lastenheftes ist es, die 'Basisanforderungen' eines Produktes in hinreichend abstrakter, verbaler Form festzuhalten. Die Anforderungen sind nummeriert, so dass auf sie zu einem späteren Zeitpunkt des Entwicklungsprozesses Bezug genommen werden kann. Das Lastenheft beschreibt das 'Was' und nicht das 'Wie' der Produkteigenschaften.

Lastenheft Retrieval Component Integrator V0.1

1. Zielbestimmung

Das Produkt soll den Anschluss multipler IR-Systeme ermöglichen, um sie in eine Metasuche einzubeziehen. Durch die Bereitstellung eines Grundgerüsts soll der Anschluss weiterer IR-Komponenten, die zum Retrievalprozess beitragen können sowie Benutzer-Applikationen geregelt werden. Das Produkt soll durch die Kombination unterschiedlicher Komponenten den Nachbau des Retrievalprozesses erlauben.

2. Produkteinsatz

Der Einsatz des Produktes ist auf heterogene IT-Infrastrukturen ausgerichtet, in denen 'Informationslieferanten' wie Datenbanken, IR-Systeme und Dokumentenmanagementsysteme mit 'Informationskonsumenten' in Form von datenverarbeitender Software gekoppelt werden sollen. Das Produkt soll zunächst im Fachbereich III der Universität Hildesheim eingesetzt werden, wo es zu Forschungszwecken die Evaluierung von IR-Verfahren und -Systemen unterstützen soll.

3. Produktfunktionen

/LF10/ Entgegennahme und Beantwortung von Anfragen.

/LF20/ Verteilung der Anfragen an heterogene IR-Systeme.

/LF30/ Verwaltung der Schnittstellen zu IR-Systemen und Benutzersoftware mit Hilfe von Plug-In-Funktionen.

/LF40/ Bereitstellung zusätzlicher Schnittstellen zu IR-Komponenten

4. Produktdaten

/LD10/ Es sind Daten zu speichern, die sich auf die technische Umsetzung der Schnittstellen beziehen. Dazu gehören a) Informationen, wie die Anfrageschnittstellen angesprochen werden können (Protokoll, Portnummer, etc.) und b) Informationen zur Anbindung der IR-Systeme (Benutzer, Passwörter, Rechner, Portnummer, etc.)

/LD20/ Speicherung von Daten, die der Identifikation und Beschreibung der IR-Systeme dienen (z.B. eindeutige ID, unterstützte IR-Strategien, etc.)

4. Produktleistungen

LL10/ Für die Funktionen /LF10/ und /LF20/ sind Mechanismen zur Parallelverarbeitung einzusetzen, so dass möglichst viele Anfragen parallel entgegengenommen und verarbeitet werden können.

5. Qualitätsanforderungen

Noten: 1 = sehr gut, 2 = gut, 3 = normal, 4 = nicht relevant.

Funktionalität:	2
Zuverlässigkeit:	3
Benutzbarkeit:	2
Effizienz:	2
Erweiterbarkeit:	1

6. Ergänzungen

Das Produkt soll funktionsfähig sein und es sollen Beispielimplementierungen einiger realer Szenarien für Testläufe zur Verfügung gestellt werden.

7 Modellierung

Dieses Kapitel beschäftigt sich mit der Modellierung realer Gegenstände oder Beziehungen, die innerhalb der Architektur von RECOIN eine Rolle spielen. Zum Ziele der Abstraktion werden verschiedene Modellierungsmethoden eingesetzt.

In diesem und dem folgenden Kapitel wird bereits auf konkrete Objekte der RECOIN-Applikation eingegangen. Zur besseren Übersicht werden sie wenn möglich direkt mit ihren Java-Klassennamen benannt. Der Bezug zu Ausdrücken der Programmiersprache Java wird durch die Verwendung des Schrifttyps `Monospace` kenntlich gemacht.

7.1 Modellierungsmethoden und -werkzeuge

> "Die Erstellung von Datenmodellen stellt eine Vorsaussetzung zur effizienten Verwaltung großer Datenbestände dar. Eine gängige Modellierungsmethode ist hier die *Entitiy Relationship* (ER)-Modellierung bzw. *Erweiterte Entity Relationship* (EER)-Modellierung. Eine erweiterte Modellierungssprache, die auch eine funktionsorientierte Modellierung ermöglicht, ist die *Unified Modelling Language* (UML)." [VOSS; GUTENSCHWAGER 2001, S. 152]

Bei einem Modell, das ganz oder in Teilen unter objektorientierten Gesichtspunkten erstellt wird, ist in jedem Fall eine Modellierungssprache wie die UML vorzuziehen, die die Objektorientierung vollständig unterstützt. Die UML stellt auch die für diese Arbeit verwendete Modellierungssprache dar. Welche Methoden und Hilfsmittel für die Modellierung eingesetzt wurden, soll im folgenden kurz vorgestellt werden.

7.1.1 Rational Unified Process und UML

Der *Rational Unified Process* (RUP) ist ein Produkt der Firma Rational Software[1], das einen *Software Engineering* Prozess darstellt, der sich durch eine iterative inkrementelle Entwicklung (cf. [QUATRANI 2000, S. 7ff.]) des Endprodukts auszeichnet. Das Produkt besteht aus spezieller Software, Anleitungen und Richtlinien, deren Einsatz die Umsetzung des RUP in Unternehmen und Projekten unterstützen. Bekannt geworden ist die Rational Software unter anderem auch durch seine Mitarbeiter wie Grady Booch, der maßgeblich an der Entwicklung der UML beteiligt war.

Die UML (*Unified Modeling Language*) ist eine einheitliche graphische Beschreibungssprache, mit der sich unterschiedliche Modellierungsmethoden in einem Rahmenkonzept integrieren lassen. Dies ist notwendig, um z.B. unterschiedliche Modelle, die verschiedene Sichten auf ein Objekt oder eine Architektur repräsentieren, miteinander zu verknüpfen.

Die UML basiert auf der Objektorientierung und bietet eine einheitliche Notation, mit der sich eine Vielzahl von Modellierungsmethoden (statische und dynamische) umsetzen lässt. Die in der UML verwendeten Diagrammtypen eignen sich darüber hinaus dazu, die einzelnen Phasen eines Entwicklungsprojekts zu begleiten. [BALZERT 1999, S. 138] sowie [VOSS; GUTENSCHWAGER 2001] behandeln die folgenden Diagrammtypen als Teil der UML:

- Anwendungsfalldiagramme (*Use Case Diagrams*)
- Aktivitätsdiagramme (*Activity Diagrams*)
- Klassendiagramme (*Class Diagrams*)
- Zustandsdiagramme (*State Charts*)
- Sequenz- und Kolaborationsdiagramme (*Sequence and Collaboration Diagrams*)
- Komponenten- und Verteilungsdiagramme (*Component and Deployment Diagrams*)

Zur Veranschaulichung von Zusammenhängen wurden für diese Arbeit Klassendiagramme sowie Kollaborations- und Sequenzdiagramme ausgewählt, deren Funktion einführend vorgestellt werden soll.

[QUATRANI 2000] erläutert, dass Kollaborations- und Sequenzdiagramme im Grunde

[1] http://www.rational.com (Zugriff: 23-Juni-2002)

genommen lediglich zwei verschiedene Sichtweisen auf ein und dasselbe Szenario darstellen. Sequenzdiagramme heben die zeitliche Abfolge von Ereignissen in den Vordergrund – was passiert als erstes, was danach, usw. Sie werden häufiger in der Analysephase eines Projekts eingesetzt, da sie aufgrund ihrer einfachen Verständlichkeit auch von eventuellen Kunden leicht interpretiert werden können. Kollaborationsdiagramme dagegen stellen einen größeren Gesamtzusammenhang her, indem sie die Interaktionen in einem Szenario auf Basis der Verbindungen der Objekte untereinander darstellen. Diese Art der Diagramme wird vermehrt in der Designphase des Projekts eingesetzt wenn die Umsetzung der Objektbeziehungen entworfen wird.

Klassendiagramme stellen eine Weiterentwicklung der in der Datenbankentwicklung und -modellierung gebräuchlichen ER- bzw. EER-Modellierung dar. Sie beschreiben eine statische Struktur, in der Klassen und Objekte eines Systems und ihre Beziehungen zueinander dargestellt werden. Der Detaillierungsgrad kann im Laufe des Entwicklungsprozesses schrittweise erweitert werden, so dass in der Anforderungsphase lediglich der Name der Klasse bekannt sein muss, um mit ihr zu arbeiten. Beim (iterativen) Durchlaufen der Phasen des Entwicklungsprozesses erhöht sich die Detailliertheit zunehmend.

7.1.2 Verwendete Modellierungs- und Entwicklungswerkzeuge

XML Spy

XML Spy ist eine Produktfamilie der Firma Altova, die den Umgang, die Erstellung und die Modellierung von XML-Dateien in der Software-Entwicklung erleichtern. Die einzelnen Komponenten dieser Familie sind ein *Integrated Development Environment* (IDE), ein XSLT-Designer[2] sowie ein Dokument-Editor und Browser Plug-Ins. XML Spy 4.4 wurde in einer Evaluierungsversion[3] eingesetzt und diente in dieser Arbeit der Modellierung und Erstellung der Schemata des Metadatenmodells in RECOIN.

Rational Rose

Eine Komponente der RUP-Software-Suite (s.o.) ist Rational Rose, ein Design-Tool, das bereits mehrfach preisausgezeichnet wurde. Es wurde für die Modellierung in dieser Ar-

[2] XSLT = Extensible Stylesheet Language Transformations, http://www.w3.org/Style/XSL/ (Zugriff: 25-Juli-2002)

[3] Eine zeitlich befristete Evaluierungsversion ist unter folgender URL erhältlich: http://www.xmlspy.com/download.html (Zugriff: 18-Juli-2002)

beit eingesetzt, da es über weitreichende Funktionen zur Generierung von Programm-Code aus Modellen verfügt. Über diese Funktionen lassen sich Daten- und Softwaremodelle in Gerüste für eine Reihe von Programmiersprachen oder z.B. in relationale DBMS-Schemata transformieren. In einem umgekehrten Schritt ist für die unterstützten Programmiersprachen auch das sog. *Reverse Engineering* von Programm-Code in ein Modell möglich. Die Entwicklung von RECOIN in der Programmiersprache Java fand auf diese Weise in einer Reihe von Durchläufen statt, bestehend aus dem Wechsel zwischen Modellierung und Reverse Engineering. Die meisten der folgenden Abbildungen stammen aus diesem Modell und entsprechen der UML-Notation. Das Programm wurde in einer Evaluierungsversion[4] verwendet.

Forte For Java

Zur Entwicklung des Java-Codes für RECOIN wurde sich für die Verwendung der Forte For Java 3.0[5] IDE entschieden. Diese IDE ist in einer speziellen Version frei erhältlich[6] und erlaubt die Integration einer Vielzahl von Hilfsmitteln, von denen hier CVS (*Concurrent Versions System*) und *Ant* eingesetzt wurden.

CVS dient der Versionskontrolle von Quelldateien und besitzt Mechanismen, die verhindern können, dass mehrere Entwickler gegenseitig ihre Änderungen an einer Datei überschreiben.

Ant[7] ist ein in Java geschriebenes *Build Tool*, das in seiner Funktion mit dem *make*-Programm auf Unix-ähnlichen Plattformen verglichen werden kann. Mit Hilfe einer in XML geschriebenen Konfigurationsdatei lassen sich mit Ant alle Schritte ausführen, um aus dem Quellcode eines Programms eine lauffähige Applikation zu erzeugen.

Sourceforge

Sourceforge[8] ist die größte und bekannteste Plattform für Open Source Entwicklungen im Internet. Sie bietet verschiedenste Services rund um die Betreuung von Projekten wie z.B. Mailing-Listen, News-Foren und auch CVS-Repositories. An der Projektmitarbeit interessierte Entwickler könne über diese Plattform direkt Kontakt mit den Projekt-Administratoren aufnehmen. RECOIN wurde dort als Projekt[9] eingetragen, um die Wei-

[4] Eine Vollversion von Rational Rose zur zeitlich befristeten Evaluierung ist erhältlich unter der URL http://www.rationalsoftware.com/tryit/rose/index.jsp (Zugriff: 06-Mai-2002)

[5] Das Produkt ist in der neuen Version 4.0 umbenannt worden in Sun ONE Studio 4.

[6] Forte For Java ist in Form einer *Community Edition* unter folgender URL frei erhältlich: http://wwws.sun.com/software/sundev/jde/buy/index.html (Zugriff: 28-April-2002)

[7] http://jakarta.apache.org/ant/, (Zugriff: 20-August-2002)

[8] http://sourceforge.net/, (Zugriff: 24- August -2002)

[9] http://recoin.sourceforge.net/, (Zugriff: 31- August-2002)

terentwicklung an einen zentralen Ort zu verlagern und somit die Verfügbarkeit der Quellen und der Dokumentation sicherzustellen.

7.2 Modellierung der Metadaten

> "Aus den dargestellten Zusammenhängen der Daten ergibt sich auch der Zusammenhang von (bereits existenten) Einzelanwendungen. Die Datenmodellierung bewirkt somit bereits einen Integrationseffekt der Daten [...] und kann zur Definition von Schnittstellen der Einzelanwendungen herangezogen werden. [...]" [VOSS; GUTENSCHWAGER 2001, S. 158]

Unter *Datenintegration* wird nun der Zugriff unterschiedlicher Software-Komponenten, inner- und außerhalb von RECOIN, auf dieselbe Datenbasis verstanden.

> "Der Stilbruch, der bei der Speicherung von in Form von Objekten vorliegenden Daten in relationalen Tabellen offenkundig wird, ist nur schwer zu leugnen. Eine Operation objekt.speichere_dich() erfordert bei Einsatz einer relationalen Datenbank entweder Handarbeit bei der Abbildung der Attribute auf Tabellenspalten oder den Einsatz eines der zahlreich verfügbaren Mapping-Werkzeuge für diese Aufgabe." [XML-Magazin, Ausg. 5/02, S. 76]

Obwohl eine Speicherung der Metadaten in einem relationalen Modell möglich wäre, wird eine objektorientiert Speicherung in Form von XML-Dateien zur Modellierung und Ablage der Metadaten vorgezogen. Daraus ergeben sich in Bezug auf die Bearbeitung und Erweiterung große Vorteile, da man nicht auf eine Datenbankschnittstelle angewiesen ist, sondern die Änderungen einfach per Text-Editor durchführen kann. Für den Fall, dass die Übertragung der Metadaten auf XML-Nachrichten basiert, entfällt außerdem eine aufwendige Umformatierung.

Das Metadatenmodell hat die Aufgabe die zu integrierenden IR-Komponenten, die beteiligten Objekte und ihre Charakteristika zu beschreiben. Die folgenden Objekte werden innerhalb von RECOIN verwendet und eignen sich deshalb als Kandidaten für die Aufnahme in das Metadatenschema. Die weiteren Angaben zu jedem Objekt spezifizieren mögliche Kandidaten für Metadaten, mit denen sich das Objekt näher beschreiben lassen könnte.

- Informationsobjekte
  - Wissensobjekt (Text, Fakten, Mischform, cf. Abb. 7.1und [WOMSER-HACKER 1996, S. 195, Abb. 7.6])
  - Dateiformat (bei Text: strukturiert, unstrukturiert)
  - Dokumentlänge
- Query-Objekt
  - QueryLanguage
- IR-System
  - IR-Modell
  - Ranking-Algorithmus
  - Kollektionsgröße
- IR-Komponente
  - Thesaurus, Stemmer, etc.

Abbildung 7.1 zeigt den grundsätzlichen Aufbau des Schemas für Informationsobjekte. In dieser Abbildung wird auch die vorgeschlagene Unterscheidung der Metadaten in RECOIN-interne (*RECOINData*) und -externe Metadaten (*IRData*) deutlich.

7.3 Modellierung von RECOIN

In diesem Kapitel werden die grundlegenden Objekte und Aufgaben modelliert, die in RECOIN implementiert werden sollen. Anhand von Kollaborations- und Klassendiagrammen sollen Objekte und Vorgänge veranschaulicht werden. Um das grundsätzliche Verständnis der dargestellten Sachverhalte zu fördern, sind die Diagramme überschaubar gehalten. Weitere Diagramme, die eine größere Menge an Objekten enthalten und komplexere Beziehungen darstellen, finden sich auf der beiliegenden CD.

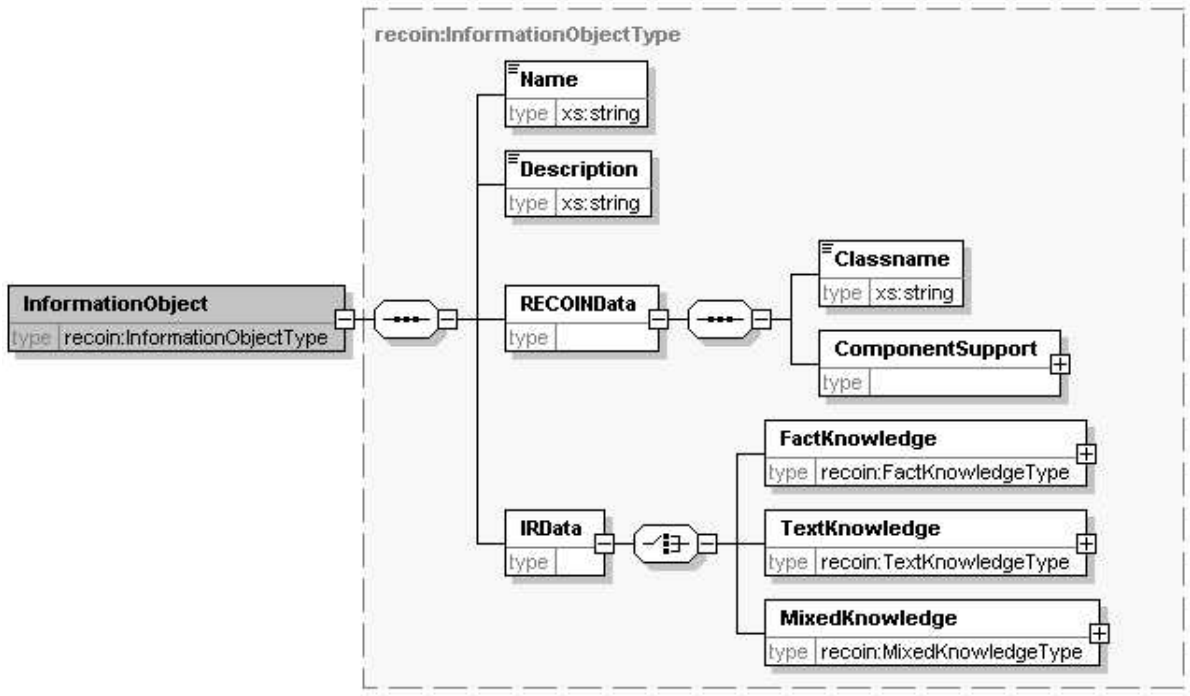

Abb. 7.1: Metadaten Schema eines Informationsobjekts mit Unterscheidung nach Wissensobjekt

7.3.1 Identifikation von Objekten

Zunächst werden die wichtigsten Objekte identifiziert, die in RECOIN realisiert werden sollen.

1. `Query`-Objekte
   `Query`-Objekte repräsentieren das Informationsbedürfnis des Benutzers und sind in einer Query Language ausgedrückt.

2. `Result`-Objekte
   `Result`-Objekte sind die Repräsentanten der Informationsobjekte, bzw. dienen als Verweise auf sie. Ein `Result` ist Bestandteil einer `ResultList`.

> "The retrieval unit is the basic element which can be retrieved as an answer to a query (normally a set of such units is retrieved [...]). The retrieval unit can be a file, a document, a web page, a paragraph or some other structural unit which contains an answer to the search query". [BAEZA-YATES; RIBEIRO-NETO 1999, S. 100]

3. `ResultList`-Objekte
   Diese Objekte enthalten eine Reihe von `Result`-Objekten und repräsentieren die Treffermenge, die von einem IR-System als Antwort auf ein `Query` zurückgeliefert wurde.

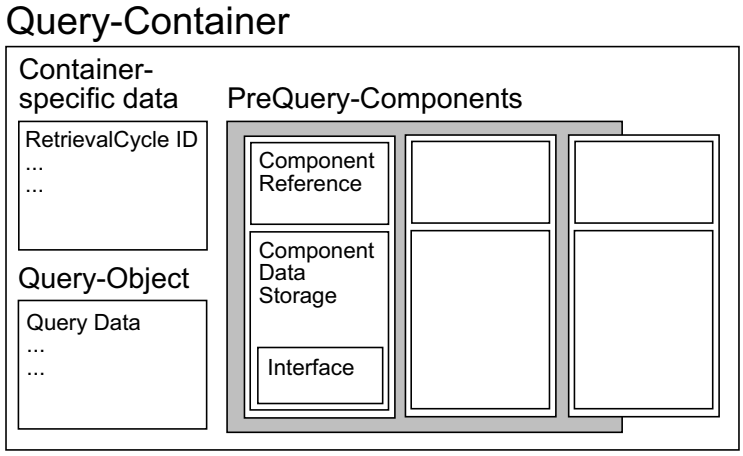

Abb. 7.2: Skizzierter Aufbau eines QueryContainer

4. `IRContainer`-Objekte
   Ein `IRContainer` dient der Kapselung von Daten innerhalb von RECOIN. Es wird eine Unterscheidung in zwei Typen vorgenommen, die durch ihre Verwendung im PreQuery- oder PostResult-Prozess motiviert ist. Damit während des Retrievalprozesses anfallende Daten unkompliziert abgelegt werden können, bieten `IRContainer`-Objekte die Möglichkeit, neue 'Unterschubladen' (`ComponentDataStorage`-Objekte) in einem Depot anzulegen. Eine Thesaurus-Komponente kann hier z.B. gefundene Synonyme oder eine Highlighting-Komponente die modifizierten Texte ablegen. Abbildung 7.2 skizziert den Aufbau am Beispiel eines `QueryContainer` und Abb. 7.3 stellt ein UML-Klassendiagramm dar, das die Zusammenhänge der bisher identifizierten Objekte beschreibt.

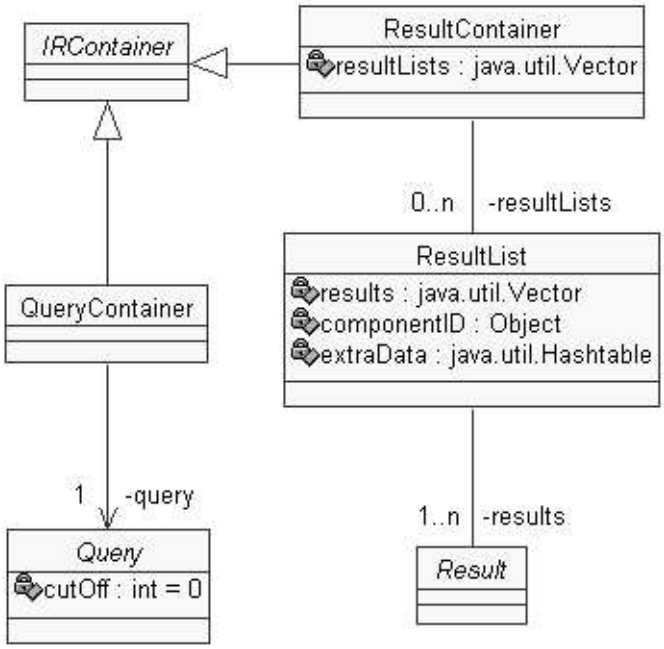

Abb. 7.3: QueryContainer und ResultContainer im Klassendiagramm

5. `IRComponent`-Objekte
   Die `IRComponent`-Objekte stellen ihre Funktionalitäten im Retrievalprozess zur Verfügung. Zu ihnen zählen beispielsweise die `Adapter`-Objekte zum Einbinden von IR-Systemen oder zusätzliche Komponenten wie Stemmer, Fusions- oder Highlighting-Komponenten, etc. Anhand ihrer Rolle im Retrievalprozess lassen sich folgende Objekte unterscheiden:

   a) `Connector`-Objekte
      `Connector`-Objekte sind die Bindeglieder zwischen RECOIN und der Anwendungssoftware. Ein `Connector` übernimmt die Entgegennahme eines Informationsbedürfnisses, das in einem speziellen Format, i.e. Protokoll, übermittelt wurde und wandelt es in ein `Query` um, falls dies nicht bereits durch die Anwendersoftware geschehen ist. Die Rückgabe von Ergebnissen erfolgt ebenfalls über den `Connector`. Zusätzlich kann an dieser Stelle eine Umwandlung der `ResultList`-Objekte in ein von der Anwendung gefordertes Format vorgenommen werden.

   b) `Adapter`-Objekte
      Die Aufgabe der `Adapter`-Objekte ist es, den Inhalt eines `QueryContainer` auszulesen und so aufzubereiten, dass ein Matching in einem angeschlossenen IR-System möglich ist. Zurückgelieferte

Resultate, wie z.b. Rankings, werden wieder in einen `ResultContainer` eingefügt und weitergereicht.

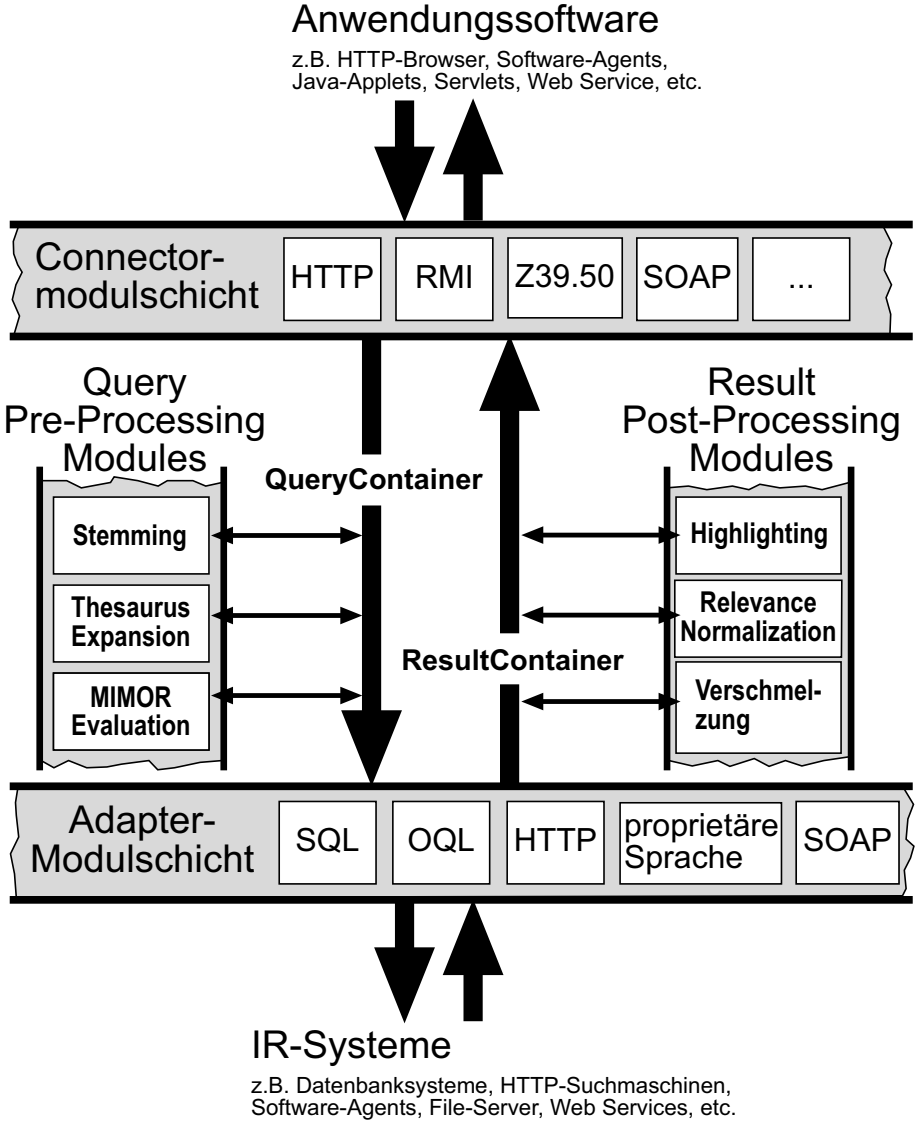

Abb. 7.4: Zusammenhang der Modulschichten in RECOIN

c) `PreQueryComponent`- und `PostResultComponent`-Objekte
Diese Objekte stellen zusätzlichen Operationen zur Verfügung, die in den Retrievalprozess integriert werden können. `PreQueryProcessor`-Objekte beschäftigen sich i.d.R. mit dem `Query` und dienen der Erweiterung oder Normalisierung während `PostResultComponent`-Objekte es erlauben, die Ergebnisse noch einmal aufzubereiten oder zu modifizieren, bevor sie über einen `Connector` wieder zurück an die Anwendungssoftware gelangen. Diese Operationen erzeugen in vielen Fällen zusätzliche Daten, die dem `IRContainer` hinzugefügt werden.

Die einzelnen `IRComponent`-Objekte werden innerhalb von RECOIN in entsprechenden `Module`-Objekten gekapselt (cf. DynamicLoading). Eine Übersicht über die so entstehenden Module-Arten und ihre Einordnung in Modulschichten entsprechend ihrer Rolle im Retrievalprozess liefert Abb. 7.4. Die beiden Enden des Containerstroms, i.e. die `Connector`- und `Adapter`-Module operieren dabei wie eine Art duale Filter, die aus externen Protokollen und Anfragesprachen `IRContainer`-Objekte erzeugen und auf dem umgekehrten Wege den Inhalt der `IRContainer`-Objekte wieder zurückwandeln. Auf diese Weise lassen sich auf beiden Strecken – vom Informationssystem zur Anwendung und umgekehrt – alle benötigten Objekte innerhalb der `IRContainer`-Objekte unterbringen, die an den Ein- und Austrittspunkten in RECOIN benötigt werden.

7.3.2 ComponentSupport-Interfaces

Ein `IRComponent` verarbeitet die in einem `IRContainer` enthaltenen Daten. Anhand der Beschreibung der `IRContainer`-Objekte im vorangegangenen Abschnitt kann man erkennen, dass es sich dabei höchstens um drei verschiedene Objekte handeln kann:

1. `Query`
2. `Result`
3. `ComponentDataStorage`

Damit `IRComponent`-Objekte diese Objekte verarbeiten können, müssen sie wissen, wie sie auf die in ihnen enthaltenen Daten zugreifen sollen. Hierfür geben sie bestimmte Schnittstellen (*Interfaces*) vor, die von den drei oben genannten Objekten implementiert werden müssen, um ihre Verarbeitung in einem `IRComponent` zu gewährleisten.

Abbildung 7.5 zeigt ein Objekt der Klasse `KeywordQuery`, das das Interface `ThesaurusSupport` implementiert. Daneben ist ein ein spezielles Objekt der Klasse `ThesaurusDataStorage` abgebildet, das das Interface `ThesaurusExpansionSupport` implementiert. Wie man in der Abbildung weiter sehen kann, implementiert die ThesaurusDataStorage ebenfalls ein Interface, nämlich ThesaurusExpansionSupport. Dieses Interface kann zu einem späteren Zeitpunkt des Retrieval Cycle von einem Adapter dazu verwendet werden, ein zusätzliches Matching mit den enthaltenen Synonymen durchzuführen.

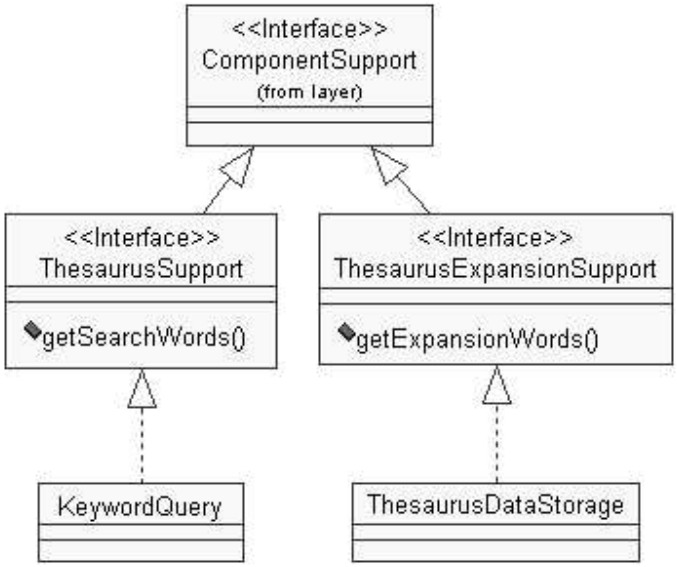

Abb. 7.5: Implementierung von ComponentSupport-Interfaces am Beispiel von KeywordQuery und ThesaurusDataStorage

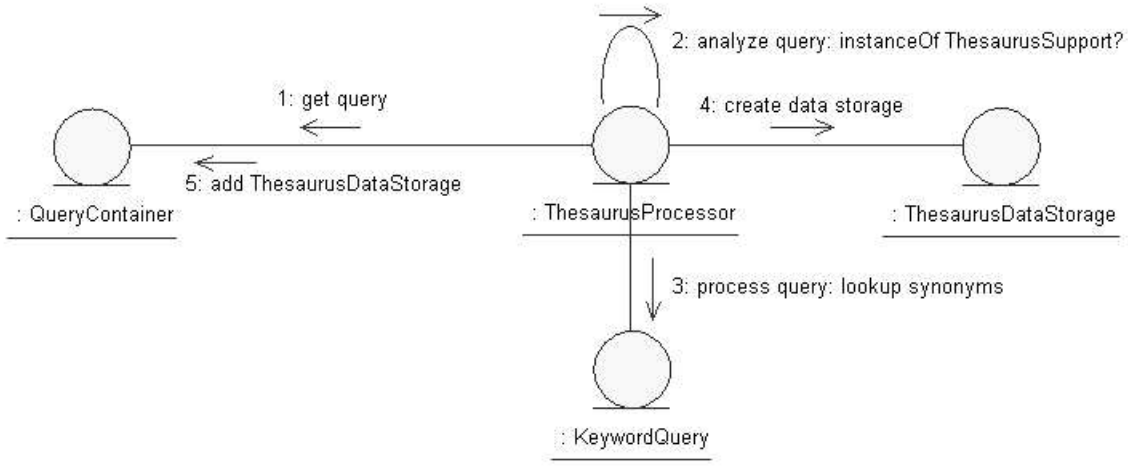

Abb. 7.6: Ablauf einer Query Expansion durch Thesaurus-Erweiterung

Wie nun das Zusammenspiel dieser Objekte mit einer Thesaurus-Komponente aussieht, verdeutlicht Abb. 7.6. Der `ThesaurusProcessor` erwartet Objekte, die das Interface `ThesaurusSupport` implementieren, über welches die nachzuschlagenden Worte extrahiert werden können. Diese Art der Verwendung von Interfaces ist ein wesentlicher Bestandteil von RECOIN. Auf der einen Seite wird durch die Interfaces der Zugriff auf die heterogenen Daten geregelt. Auf der anderen Seite werden die Interfaces als Attribute eines `IRComponent` mit in der Metadatenbank verwaltet. Mit Hilfe dieser Angaben lässt sich so vor dem Start eines `RetrievalCycle` rekursiv ermitteln, welche `IRComponent`-Objekte bei Verwendung welches `Query` für eine Einbindung in den Cycle überhaupt in Frage kommen.

7.3.3 RetrievalSession und RetrievalCycle

Der `RetrievalCycle` bestimmt, welche Objekte in einen konkreten Retrievalprozess eingebunden werden sollen. Er ist die aktive Steuereinheit für die Abläufe im Retrievalprozess und ist in der Lage, den `IRContainer` an die Stationen (`ModuleLayer`-Objekte), in denen er bearbeitet werden soll, weiterzureichen.

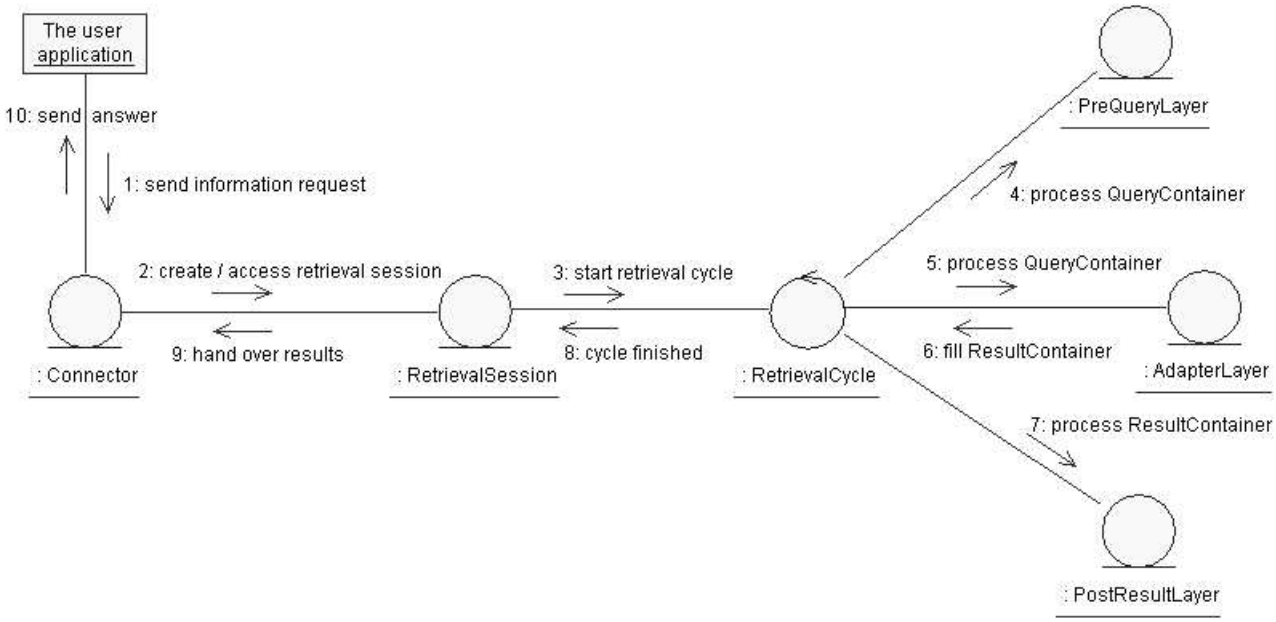

Abb. 7.7: Kollaborationsdiagramm für den Ablauf einer RetrievalSession

Wann die Verarbeitung eines `IRContainer` abgeschlossen ist und dieser weitergeleitet werden soll, wird durch einen Trigger-Mechanismus bestimmt, der darauf basiert, den `RetrievalCycle` so lange zu blockieren bis der `IRContainer` ein Signal gibt, dass seine Bearbeitung abgeschlossen ist. Abbildung 7.7 Abb:RetrievalSession stellt den grundlegenden Ablauf einer `RetrievalSession` als Kollaborationsdiagramm dar.

7.3.4 Nebenläufigkeit durch multiple Sessions und Worker-Threads

Zur Sicherstellung der gleichzeitigen Nutzung von RECOIN durch mehrere Benutzer oder Applikationen werden Anfragen an RECOIN, in `RetrievalSession`-Objekten gekapselt. Die einzelnen Durchläufe (Cycles) der Session sind als eigenständige Threads, sog. *Lightweight-Prozesse* (cf. [LEA 2000, S. 34]), implementiert. Dabei ist zu beobachten, dass ein `RetrievalCycle` selber eine Reihe von Aufgaben erledigen muss, die parallel ausgeführt werden können. Die Mehrheit dieser Aufgaben steht in Verbindung zu der möglichen gleichzeitigen Bearbeitung eines `IRContainer` durch mehrere `IRComponent`-Objekte, zu denen auch die `Adapter`-Objekte der IR-Systeme zählen.

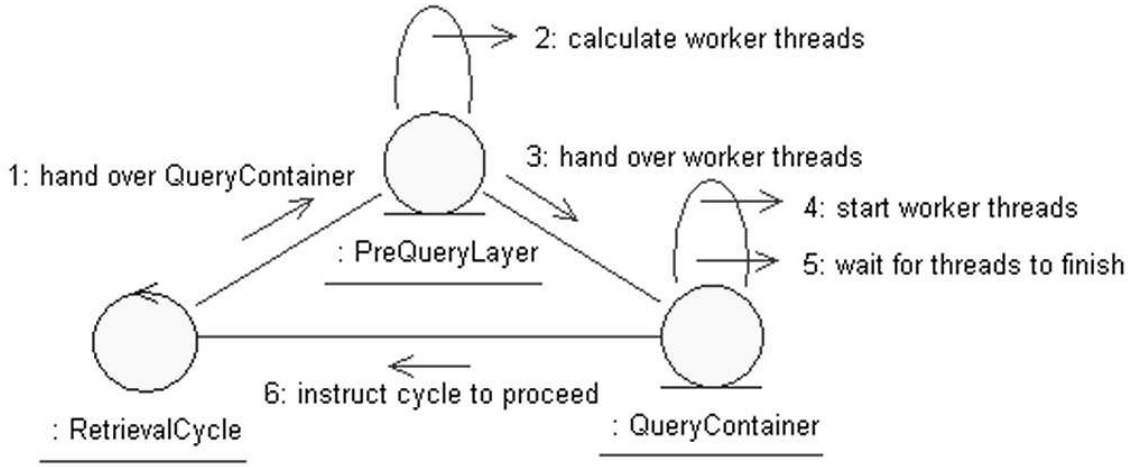

Abb. 7.8: Die Bearbeitung eines IRContainer durch mehrere unabhängige Worker-Threads am Beispiel des PreQueryLayer

Zur Implementierung dieser Nebenläufigkeit werden sog. *Worker-Threads* (cf. [LEA 2000, S. 290ff.]) eingesetzt, die von einem `IRComponent` erzeugt und dann vom `IRContainer` gestartet werden können. Der Vorteil dieser Umsetzung liegt darin, dass der Container jederzeit den Überblick hat, welche Objekte noch an ihm arbeiten. Durch Beendigung aller Worker-Threads wird der bereits in Kap. 7.3.3 beschriebene Trigger ausgelöst, der die `RetrievalSession` veranlasst, den `IRContainer` weiterzureichen. Abbildung 7.8 veranschaulicht diesen Mechanismus. Durch das hohe Maß an Nebenläufigkeit und einer gleichzeitigen Nutzung knapper Ressourcen in Form von `IRComponent`-Objekten werden grundlegende Sicherheitsmechanismen erforderlich. Beim Zugriff auf Attribute und Methoden, die von mehreren Threads gleichzeitig genutzt (und verändert) werden können, werden deshalb kritische Bereiche durch *Synchronisationsmechanismen* geschützt, die wie das Schlüsselwort `synchronized` Bestandteil der Programmiersprache sind. Die Zugriff auf geschützten Teile eines Objekts ist solange blockiert, bis ein `Thread` das Objekt durch Beendigung seiner Aufgabe oder Abbruch wieder freigibt. Dies ist z.B. notwendig, wenn zur Laufzeit die `Module`-Objekte eines `ModuleLayer` erneuert werden sollen. Solange der `ModuleLayer` noch von einem `RetrievalCycle` verwendet wird, muss der Update-Mechanismus warten.

8 Entwicklung

Im vorangegangenen Kap. 7.3 sind bereits eine Vielzahl konkreter Objekte benannt worden, die in RECOIN eine Rolle spielen. Eine vollständige Diskussion aller Objekte und Klassen, die den Funktionsumfang von RECOIN ausmachen, wird an dieser Stelle allerdings als wenig sinnvoll erachtet; für einen tieferen Einblick in die einzelnen Klassen wird auf die Java-Dokumentation auf der beiliegenden CD verwiesen (cf. Anhang). Auf der CD finden sich außerdem weitere UML-Diagramme, die aufgrund ihres Umfangs oder ihrer Komplexität an dieser Stelle nicht dargestellt werden können.

In diesem Kapitel sollen wichtige Konzepte, die bei der Entwicklung und Implementierung von RECOIN eine Rolle spielen, vorgestellt werden. Dies soll vor allem den Einstieg in die Erweiterung von RECOIN durch die Programmierung eigener Klassen erleichtern.

8.1 Erweiterbarkeit durch abstrakte Basisklassen

RECOIN stellt zur Erweiterung des Funktionsumfangs eine Reihe von Basisklassen zur Verfügung. Jeder Entwickler oder Programmierer kann mit ihrer Hilfe eigene Objekte entwerfen und entwickeln, indem er von den Basisklassen erbt. Dies muss er sogar, da die Basisklassen abstrakte Klassen sind, die den generellen Aufbau eines Objekts vorschreiben, selber aber nicht instanziiert werden können (cf. [FLANAGAN 1997, S. 75f.]). Zu den wichtigsten Basisklassen zählen:

- `Query`
- `Result`
- `ResultList`
- `IRComponent` (`Connector`, `Adapter`, etc.)

Das folgende Beispiel soll verdeutlichen, wie die Einbindung neuer Funktionen in RECOIN durch Vererbung vereinfacht wird.

Angenommen, man möchte in RECOIN eine weitere Datenbank integrieren, so ist es zunächst notwendig, eine eigene Klasse zu entwickeln, die die Kommunikation mit dem DBMS übernimmt und die von der Basisklasse `Adapter` abgeleitet ist. Verlangt der `Adapter` nach einem speziellen `Query` oder erzeugt er `Result`-Objekte, die noch nicht Teil des Funktionsumfangs von RECOIN sind, so ist es weiter notwendig, diese durch Beererbung der Klassen `Query` und `Result` bereitzustellen. Zur Verwendung der neuen Funktionen fehlt nun noch ein `Connector`-Objekt, das die `Query`- und `Result`-Objekte in der Kommunikation mit einem Client verwendet.

Um die Einbindung in RECOIN abzuschließen, müssen alle neuen Klassen der Paketstruktur hinzugefügt und ihre Attribute und Funktionen in den Metadaten deklariert werden. Als oberstes Paket für alle Erweiterungen in Form von zusätzlichen Klassen sollte das Paket recoinx (*RECOIN eXtensions*) verwendet werden[1].

8.2 Module-Objekte und dynamisches Laden

Module-Objekte dienen in RECOIN als Behältnisse für `IRComponent`-Objekte. Durch die Kapselung in Modulen lässt sich das Hinzufügen und Entfernen der `IRComponent`-Objekte und ihrer Funktionen vereinheitlichen. Um diese Funktionen zur Laufzeit einzubinden, unterstützen `Module`-Objekte die dynamische Instanziierung der `IRComoponent`-Objekte. Der hierfür benötigte Klassenname des konkreten `IRComponent` wird durch einen Eintrag in den Metadaten des Objekts bekannt gegeben. Dieser Mechanismus zum dynamischen Laden von Objekten wird in RECOIN durchgehend angewandt. Es werden z.B. auch die speziellen `ComponentDataStorage`-Objekte, die einem `IRComponent` zur Ablage von Daten im `IRContainer` dienen, anhand ihres Klassennamens erzeugt, wenn sie gebraucht werden.

In einem Module können auch mehrere `IRComponent`-Objekte untergebracht werden, falls sie z.B. Bestandteile einer komplexen Aufgabe sind oder sich eine gemeinsame Ressource teilen. So können beispielsweise die offenen Verbindungen zu einem DBMS in einem `ConnectionPool` implementiert werden, so dass sich die im `Module` realisierten `Adapter` diese teilen können[2].

[1] Für einen Überblick über die Klassen und die Paketstruktur von RECOIN sei auf die Java-Dokumentation auf der beiliegenden CD verwiesen.

[2] Das JDBC 3.0 (*Java Database Connectivity*) erlaubt außer dem *Connection Sharing* zusätzlich die gemeinsame Nutzung vorkompilierter Anweisungen, sog. *Prepared Statements*, in einem Pool (cf. [GUTJAHR; LOEW 2002]). `Adapter`, die die gleichen Statements an eine Datenbank verwenden, profitieren davon durch eine beschleunigte Ausführung.

9 Fazit und Ausblick

In dieser Arbeit sollte gezeigt werden, dass es beim Umgang mit heterogenen Datenbeständen und IR-Systemen eine Vielzahl von Hürden gibt, die verhindern, dass einzelne Anwendungen zu einem Gesamtsystem integriert werden können.

Mit RECOIN existiert ein Grundgerüst, um in einem zentralen Punkt die funktionalen Fähigkeiten von IR-Komponenten zu bündeln. Funktionen werden auf diese Weise aus den Komponenten ausgelagert und anderen Komponenten als Dienste zur Verfügung gestellt. Dieses 'Outsourcing' von Funktionalität ermöglicht eine bessere Nutzung der vorhandenen Kapazitäten.

Als weiterer wichtiger Punkt kann festgehalten werden, dass es aufgrund der Architektur der RECOIN möglich ist, ganz individuelle, einfache oder multiple Verknüpfungen zwischen IR-Komponenten herzustellen. Das bedeutet, dass es auf der einen Seite keine Probleme bereitet, eine eigene, spezialisierte Datenbank über einen selbstdefinierten Adapter anzuschließen und einen speziell auf seine Wünsche abgestimmten `Connector` zu schreiben, der ausschließlich mit diesem `Adapter` kommuniziert. Auf der anderen Seite kann aber auch ein `Connector` mit mehreren `Adapter`-Objekten oder IRComponent-Objekten zusammenarbeiten, um komplexe mehrschrittige Prozesse auszuführen.

Auf diese Weise lässt sich jeder Retrievalprozess durch eine spezielle Kombination oder Verkettung von Bausteinen (Modulen) darstellen. Genau in dieser möglicherweise dynamischen Verkettung der Bausteine liegt ein Vorteil der Modularisierung des Retrievalprozesses. Dies unterstreicht die Funktion von RECOIN zum Aufbau flexibler und erweiterbarer Retrieval Cycles.

Der meiner Meinung nach größte Vorteil besteht in der einfachen Erweiterbarkeit des Funktionsumfangs, da durch RECOIN bereits das Gerüst und fundamentale Verwaltungsfunktionen zur Verfügung gestellt werden. Mit einem minimalen Programmieraufwand ist es damit möglich, zusätzliche Objekte und Komponenten zu integrieren.

Der Entwicklungstand von RECOIN hat zu diesem Zeitpunkt die Version 0.1 erreicht und ist unter der GPL (*GNU Public License*) veröffentlicht worden. Als Projekt ist

RECOIN unter der URL http://recoin.sourceforge.net erreichbar.

Auf diese Weise soll der Zugang zu Dokumentation und Quellen von RECOIN sichergestellt werden. Hauptsächlich soll durch diese Öffentlichkeit aber auch erreicht werden, dass jeder Interessierte seine Ideen einbringen, sich aktiv beteiligen und somit die Weiterentwicklung von RECOIN unterstützen kann.

RECOIN stellt eine ausgezeichnete Basis dar, um effizient auf das Information Retrieval bezogene Funktionen zu bündeln und zu kombinieren. Es besteht deshalb die Hoffnung, durch rege Beteiligung an der Weiterentwicklung den Funktionsumfang von RECOIN so stark auszubauen, dass ein multifunktionales Baukastensystem entsteht, durch das die steigende Heterogenität der Datenbestände und Informationssysteme bewältigt werden kann.

Inhalt der CD

Auf der dieser Arbeit beiliegenden CD findet sich der folgende Inhalt:

- README-Datei
- Diese Arbeit in verschiedenen Formaten (PDF, Postscript, TEX)
- Quelldateien und Dokumentation für RECOIN
- Zum Betrieb von RECOIN notwendige Software:
  - Java Development Kit (JDK) 1.4
  - Tomcat Application-Server Version 4.04
  - Ant Build Tool Version 1.5
  - dom4j Version 1.3
  - JDBC-Treiber für MySQL und PostgreSQL

Eine detailliertere Auflistung des Inhalts sowie Anmerkungen zum Gebrauch oder zur Installation der Software enthält die README-Datei.

Eigenständigkeitserklärung

Ich versichere hiermit, dass ich die Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen oder anderen Quellen entnommen sind, sind als solche kenntlich gemacht.

Datum Unterschrift

Literaturverzeichnis

[BAEZA-YATES; RIBEIRO-NETO 1999]
Ricardo Baeza-Yates; Berthier Ribeiro Neto, *"Modern Information Retrieval"*, Addison Wesley, 1999, ISBN 0-201-39829-X

[BALZERT 1996]
Helmut Balzert, Lehrbuch der Software-Technik : Software-Entwicklung, Spektrum Verlag, Heidelberg/Berlin/Oxford, 1996, ISBN 3-8274-0042-2

[BALZERT 1999]
Helmut Balzert, Lehrbuch Grundlagen der Informatik, Spektrum Verlag, Heidelberg/Berlin, 1999, ISBN 3-8274-0358-8

[FLANAGAN 1997]
David Flanagan, *"Java in a nutshell 2nd Ed."*, O'Reilly, 1997, ISBN 1-56592-262-X

[GOLDFARB 2002]
Charles F. Goldfarb, *"Charles F. Goldfarb's XML Handbook - 4th ed., Prentice Hall"*, 2002, ISBN 0-13-065198-2

[GROSSMAN; FRIEDER 1998]
David A. Grossman; Ophir Frieder, *"Information Retrieval - Algorithms and Heuristics"*, Kluwer Academic Publishers, Boston/Dordrecht/London, 1998

[GUTJAHR; LOEW 2002]
Julia Gutjahr; Andreas Loew, *"Griff nach den Daten - Neuigkeiten im Java Database Connectivity API 3.0 und JDBC Best Practices"*, Javamavazin, Ausg. 7.2002, S. 35-44

[KEMPER; EICKLER 1999]
Alfons Kemper; Andreas Eickler, *"Datenbanksysteme - Eine Einführung"*, 3. Aufl., Oldenbourg-Verlag, München, 1999, ISBN 3-486-25053-1

[KOWALSKI 1997]
Gerald Kowalski, *"Information Retrieval Systems: Theory and Implementation"*, Kluwer Academic Publishers, Boston/Dordrecht/London, 1997

[KRCMAR 2000]
Helmut Krcmar, *"Informationsmanagement"*, 2. Aufl., Springer-Verlag, Berlin, 2000, ISBN 3-540-66359-2

[LEA 2000]
Doug Lea, "*Concurrent Programming in Java 2nd Ed - Design Patterns and Principles*", Addison Wesley, Boston, 2000, ISBN 0-201-31009-0

[LEHNER 2000]
Franz Lehner, "*Organisational Memory*", Carl Hanser Verlag, München, 2000, ISBN 3-446-21357-0

[LJUBOJEVIC 1991]
M. Ljubojevic, "*Repositories mit objektorientiertem Informationsmodell*", in: HMD Theorie und Praxis der Wirtschaftsinformatik, Heft 161, Jahrgang 28, 1991, S. 35-44.

[LONEY; KOCH 2001]
Kevin Loney; George Koch, "*Oracle 8i - Die umfassende Referenz*", Carl Hanser Verlag, München/Wien, 2001, ISBN 3-446-21570-0

[MYRACH 1994]
Thomas Myrach, "*Konzeption und Stand des Einsatzes von Data Dictionaries*", (Beiträge zur Wirtschaftsinformatik Bd. 12), Physica Verlag, Heidelberg, 1994, ISBN 3-7908-0822-9

[NAUER 1991]
Bernhard Nauer, "*Das Repository-Informationsmodell als zentrale Schnittstelle*", in: HMD Theorie und Praxis der Wirtschaftsinformatik, Heft 161, Jahrgang 28, 1991, S. 26-34.

[RUMELHART; MCCLELLAND 1986]
David E. Rumelhart; James L. McClelland, "*Parallel Distributed Processing. Explorations in the Microstructure of Cognition, Volume 1: Foundations*", MIT Press, Massachusetts, 1992

[SALTON; MCGILL 1983]
Gerard Salton; Michael J. McGill, "*Information Retrieval - Grundlegendes für Informationswissenschaftler*", McGraw-Hill, New York; Hambug, 1987, ISBN 3-89028-051-X

[QUATRANI 2000]
Terry Quatrani, "*Visual modeling with Rational Rose 2000 and UML*", Addison-Wesley, 2000, ISBN 0-201-69961-3

[REIF 2000]
Gerald Reif, "*Moderne Aspekte des Wissensverarbeitung*", Diplomarbeit, Technische Universität Graz, 2000, http://www.iicm.edu/thesis/greif (Zugriff: 12-August-2002)

[VOSS; GUTENSCHWAGER 2001]
Stefan Voß, Kai Gutenschwager, "*Informationsmanagement*", Springer-Verlag, Berlin/Heidelberg, 2001, ISBN 3-540-67807-7

[WOLLNIK 1988]
M. Wollnik, "*Ein Referenzmodell des Informationsmanagements, Information Management 3*", 1988, S. 34-43

[WOMSER-HACKER 1996]
Christa Womser-Hacker, *"Das MIMOR-Modell. Mehrfachindexierung zur dynamischen Methoden-Objekt-Relationierung im Information Retrieval"*, Habilschrift, Universität Regensburg, 1996